University of Alberta

DISCOVERING AND RANKING OUTLIERS IN VERY LARGE DATASETS

by

©

Yaling Pei

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfillment of the requirements for the degree of **Master of Science**

Department of Computing Science

Edmonton, Alberta
Fall 2006

Library and
Archives Canada

Published Heritage
Branch

395 Wellington Street
Ottawa ON K1A 0N4
Canada

Bibliothèque et
Archives Canada

Direction du
Patrimoine de l'édition

395, rue Wellington
Ottawa ON K1A 0N4
Canada

Your file Votre référence
ISBN: 978-0-494-22342-0
Our file Notre référence
ISBN: 978-0-494-22342-0

NOTICE:
The author has granted a non-exclusive license allowing Library and Archives Canada to reproduce, publish, archive, preserve, conserve, communicate to the public by telecommunication or on the Internet, loan, distribute and sell theses worldwide, for commercial or non-commercial purposes, in microform, paper, electronic and/or any other formats.

The author retains copyright ownership and moral rights in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

AVIS:
L'auteur a accordé une licence non exclusive permettant à la Bibliothèque et Archives Canada de reproduire, publier, archiver, sauvegarder, conserver, transmettre au public par télécommunication ou par l'Internet, prêter, distribuer et vendre des thèses partout dans le monde, à des fins commerciales ou autres, sur support microforme, papier, électronique et/ou autres formats.

L'auteur conserve la propriété du droit d'auteur et des droits moraux qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

In compliance with the Canadian Privacy Act some supporting forms may have been removed from this thesis.

While these forms may be included in the document page count, their removal does not represent any loss of content from the thesis.

Conformément à la loi canadienne sur la protection de la vie privée, quelques formulaires secondaires ont été enlevés de cette thèse.

Bien que ces formulaires aient inclus dans la pagination, il n'y aura aucun contenu manquant.

Canada

Abstract

Outlier detection aims to discover exceptional instances in datasets. It has long been studied in the literature of statistics. In recent years, outlier detection has gained much interest in data mining and found many important applications. Current work on outlier detection mainly focuses on three aspects: definition of outliers, efficient methods for finding meaningful outliers and evaluation methodology. In this thesis, we propose a new method that uses the relative degree of density with respect to a set of reference points to estimate the neighborhood density of a data point. Candidate outliers are ranked based on the outlier score that is assigned to each data point. The running time of our reference-based algorithm is $O(Rn \log n)$ where n is the size of the dataset and R is the number of reference points. Analysis and experiments show that our method is very effective and highly scalable to very large datasets. To facilitate experimental tests for outlier analysis and automate the generation of diverse datasets, we developed a generic framework for synthetic data generation. The system can efficiently produce datasets with various characteristics such as size, shape, density as well as cluster and outlier distributions.

Acknowledgements

First of all, I would like to thank my supervisor, Dr. Osmar Zaiane, for his advice and encouragement. This thesis is by far the most significant academic accomplishment in my life and it would have been impossible without his guidance and inspiration. His deep vision, broad knowledge and constructive suggestions have guided me over the years.

I want to thank my other committee members, Dr. Jörg Sander, Dr. Vadim Bulitko, and Dr. Scott Dick, for reading my thesis draft. Their comments are invaluable for the completion of this thesis.

I wish to express my gratitude to many people with the department, in particular, to Dr. Mario A. Nascimento, Dr. Jia You, Dr. Herb Yang, Dr. Russ Greiner, Dr. Dekang Lin and Dr. Li-Yan Yuan. I sincerely thank them for their supervision and help during my study in the department. Thanks are also devoted to graduate student advisors Edith Drummond and Karen Berg. They were always there to help and made me feel so spoiled.

Thanks to Dr. Rong-Qing Jia from Department of Mathematical and Statistical Sciences for teaching me how to think and solve problems from the mathematics perspective.

I would also like to thank Natural Sciences and Engineering Research Council, iCore and University of Alberta for the support of my graduate study.

Thanks, as well, to my husband Yong Gao for many helpful discussions on our research work. I am grateful to my parents for their unconditional love and support that have been with me throughout the years.

Table of Contents

1 Introduction **1**
- 1.1 Problem Statements 3
- 1.2 Contributions 6
- 1.3 Organization of this thesis 7

2 Related Work **8**
- 2.1 Statistical Outlier Detection 9
  - 2.1.1 Graphical tools 10
  - 2.1.2 Distribution-based Outlier Detection 11
  - 2.1.3 Depth-based Outlier Detection 13
- 2.2 Outlier Detection in Clustering Analysis 14
- 2.3 Distance-based Outlier Detection 15
  - 2.3.1 Basic Concepts 15
  - 2.3.2 Algorithms for Mining Distance-based Outliers 18
  - 2.3.3 Summary of Distance-based Outlier Detection Methods . . 22
- 2.4 Local Outlier Detection 24
  - 2.4.1 Density-based Outlier Detection: Local Outlier Factor (LOF) 24
  - 2.4.2 Connectivity-based Outlier Detection: Connectivity-based Outlier Factor(COF) 27
  - 2.4.3 Algorithms for Finding Connectivity-based Local Outlier Factor 30
  - 2.4.4 Summary of Local Outlier Detection 31
- 2.5 Neural network Outlier Detection 31

3 An Efficient Reference-based Approach to Outlier Detection in Large Datasets **33**
- 3.1 Motivation 33
- 3.2 Review of Distance-based Outliers 35
- 3.3 Reference-based Outlier Detection Method 36
  - 3.3.1 Compatibility with Distance-based Method 38
  - 3.3.2 Algorithm and Its Implementation 41
  - 3.3.3 Determination of Reference Points 43
  - 3.3.4 Detecting Global and Local Outliers in Complex Datasets . 44
- 3.4 Empirical Evaluation 50
  - 3.4.1 Results on Synthetic Datasets 50
  - 3.4.2 Results on Hockey Data 54
- 3.5 Conclusion 56

4 A Synthetic Data Generator for Clustering and Outlier Analysis **58**
- 4.1 Introduction 59
- 4.2 Existing Work on Synthetic Data Generation 60
  - 4.2.1 IBM Quest Synthetic Data Generator 61

4.2.2 Synthetic Data Generation in Other Research Fields 62
4.3 Mathematical Tools and Techniques 63
4.3.1 Uniform Distribution 63
4.3.2 Normal Distribution 64
4.3.3 Box-muller Transformation 65
4.3.4 Linear Transformation 65
4.4 A Comprehensive Approach to Synthetic Data Generation 68
4.4.1 Generation of Clusters in a Dataset 71
4.4.2 Generation of Outliers/Noise in a Dataset 76
4.5 Experiments and Evaluation 79
4.5.1 Generating Very Large Datasets 80
4.5.2 Testing with Clustering and Outlier Analysis Algorithms . . 82
4.6 Conclusion 91

5 Conclusions and Future Work 93

Bibliography 95

A A GUI-based Web Application for Data Clustering and Outlier Detection 101
A.1 Goals 101
A.2 A Virtual Lab for Learning Clustering methods 103
A.3 An Interactive Web-based Testbed for Clustering Analysis 105

List of Tables

2.1 A variation of NL algorithm for finding distance-based outliers [8] . 20

3.1 Outlier detection result 1 on NHL(03/04) data 55
3.2 Shooting percentage on NHL(03/04) data 55
3.3 Outlier detection result 2 on NHL(03/04) data 56

4.1 Detailed description of the parameters for the datasets 82
4.2 Description of the clustering algorithms 82

List of Figures

1.1 Examples of different types of outliers 3

2.1 Classification of outlier detection methods in [66] 9
2.2 Example of a box plot 10
2.3 Example of a scatter plot 11
2.4 standard normal distribution 12
2.5 Some depth contours for a dataset containing 500 points 13
2.6 Example of $DB(p, D)$-outlier detection, where $p = 95\%$, and $D = 20$ 17
2.7 Failure of detecting outlier $o1$ using distance-based methods 23
2.8 Failure of Outliers detection using LOF 28
2.9 A simplified example showing outlier detection using COF 28
2.10 A schematic view of a fully connected replicator neural network . . 32

3.1 Reference-based nearest neighbors in the one-dimensional dataset X^p 37
3.2 Reference-based nearest neighbors satisfying the triangle inequality 39
3.3 A 2D dataset containing multiple clusters with local outliers 45
3.4 Finding top eight outliers (a) using reference-based method and (b) using the traditional KNN method 46
3.5 Finding top 76 outliers (a) using reference-based method and (b) using the traditional KNN method 47
3.6 Finding local outliers using reference-based approach 48
3.7 Log-scale execution time vs data size for reference-based approach and Orca 51
3.8 Outlier detection result from KNN-based approach 53
3.9 Outlier detection result from LOF 53
3.10 Outlier detection result from ROS 54

4.1 PDF and CDF of uniform distribution 63
4.2 PDF and CDF of normal distribution 64
4.3 Linear transformation: expansions and contractions 66
4.4 Linear transformation: shears 66
4.5 Linear transformation examples 67
4.6 A screen shot of the synthetic data generation system 69
4.7 Difficulty level 1: each cluster contains 500 2D points 72
4.8 Difficulty level 2: each cluster contains 300 2D points 73
4.9 Difficulty level 3: each cluster contains 500 2D points 73
4.10 Difficulty level 4: each cluster contains 500 2D points 74
4.11 Difficulty level 5: each paired cluster contains 1,000 2D data points, of which 500 are assigned to each single cluster 75
4.12 Outliers level none: outliers are those exterior points of a cluster . . 77
4.13 Outliers level low: outliers are randomly distributed 78
4.14 Outliers level high: outliers are either randomly distributed or have simple patterns 79

4.15 Each generated dataset has the following properties: number of clusters is 5; data distribution in a cluster is either uniform or normal; difficulty level ranges from 1 to 5, density level is 3, and noise level is low 81
4.16 With each difficulty level, the system generates a dataset of 100,000 that contains both uniformly and normally distributed clusters. 81
4.17 Clustering results on dataset 1. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass 84
4.18 Clustering results on dataset 2. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass 85
4.19 Clustering results on dataset 3. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass 86
4.20 Clustering results on dataset 4. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass 87
4.21 Clustering results on dataset 5. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass 88
4.22 Clustering results on dataset 6. (a): k-means with $k = 7$; (b): DBSCAN with $\epsilon = 20$ and $MinPts = 30$; (c): CURE with $k = 7$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k-NN = 15$, MinSize=2.5%, and $k = 7$; (e): WaveCluster with $r = 4$ and $\tau = 0.2$; (f): AutoClass 89

A.1 Data clustering analysis - welcome 102
A.2 Data clustering analysis - algorithms 103
A.3 Data clustering analysis - applet 104
A.4 UML of the Web-based clustering system 106
A.5 Clustering testbed introduction page 109
A.6 Clustering testbed registration page 109
A.7 Clustering testbed registration validation page 110
A.8 Clustering testbed algorithm selection page 110
A.9 Clustering testbed algorithm parameter page 111
A.10 Clustering testbed algorithm output page 111
A.11 Clustering testbed output presentation page 112

Chapter 1

Introduction

The huge amount of data available provides us with a rich source of information, but it also makes it extremely hard for human experts to gain knowledge by manually observing the raw data or using the traditional methods that are not scalable to large data sets. With the increasing demand of specialized techniques and tools for accurate data analysis, data mining has become an interdisciplinary field and gained much popularity in both research and industrial communities in the last decade. Also known as Knowledge Discovery in Databases (KDD), data mining is the process of non-trivial extraction of implicit, previously unknown, and potentially useful information from data [25].

Data mining involves various tasks of pattern discovery in large datasets. In [31], these tasks are classified into two major categories: descriptive and predictive. Descriptive mining activities aim to summarize the general properties of the data. Predictive mining activities try to make predictions based on the characteristics of the current datasets. Most data mining tasks such as classification, clustering and association rule mining attempt to identify useful data patterns implied in the majority of the dataset. Outlier analysis, on the other hand, deals with identifying a small portion of the data. This small amount of data are usually significantly different in some or all the dimensions from the remaining data points. Being an important task in data analysis, outlier detection and analysis has drawn much attention in the area of data mining in recent years. Raw data collected from real-life activities are often imperfect and contain aberrant values. These aberrant values can be noise that leads to a poor data model. To avoid the influence of such noise in the process of mining data regularities, data cleaning serves as an important step in the KDD process to eliminate unwanted data values. But in many situations, the exceptional cases are not necessarily bad data points and are often more interesting than common events. The identification of such outliers can provide valuable information not only about the data collecting and recording process but also about the abnormal activities. The goal of outlier detection is two-fold: to screen for exceptional data values, and to uncover the implicit patterns of those rare cases that contain knowledge of particular interest. Further study and analysis on the characteristics of the identified outliers are often conducted to build knowledge models of the abnormal behavior.

Outlier detection has found many real life applications, where the identification of outliers is essential in finding rare and suspicious activities. Some typical examples are fraud detection [11], for example, detecting credit card fraud and finding criminal activities in E-commerce, network intrusion identification [19, 49], video surveillance monitor [47], sports and market analysis [45, 14].

In this thesis, we will concentrate on outlier detection. We investigate different techniques and methods, discuss recent work and existing problems in outlier analysis in the area of data mining. We propose a new approach to outlier detection. The proposed outlier detection method uses the relative degree of density with respect to a set of reference points to approximate the degree of density defined in terms of the k nearest neighbors of a data point. Candidate outliers are ranked based on the reference-based outlier score that has been assigned to each data point. The worst case running time of our algorithm is $O(Rn \log n)$, where n is the size of the dataset and R is the number of reference points. Analysis and experiments demonstrate that our method is a feasible and efficient alternative for detecting distance-based outliers. In order to facilitate the evaluation of our outlier algorithm, we designed and implemented a generic framework for synthetic data generation. It can dynamically generate datasets of different probability distributions with various difficulty levels in terms of clusters and outliers.

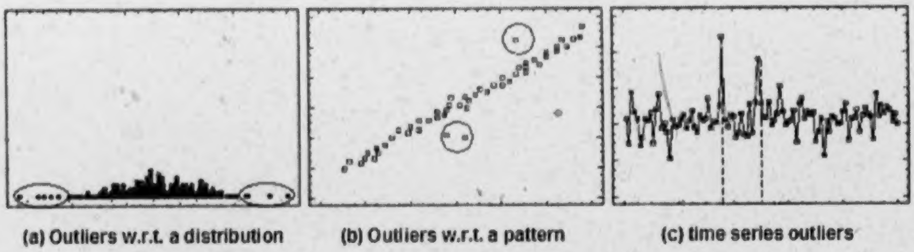

(a) Outliers w.r.t. a distribution (b) Outliers w.r.t. a pattern (c) time series outliers

Figure 1.1: Examples of different types of outliers

1.1 Problem Statements

In physics, the statistical term outlier is often included in the more general term "noise". Intuitively, an outlier is a data value that lies far away from the rest of the data. While there is no universal definition for an outlier, some widely accepted definitions given in statistics include

- *"An outlier is an observation that deviates so much from other observations as to arouse suspicions that it was generated by a different mechanism"* [33]
- *"an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data"* [6]

Although there exists some controversy over what constitutes an outlier, it is generally considered that an outlier has unusual or exceptional data value. It is an observation that appears to be inconsistent with the remainder of the dataset. Outliers are often interesting observations which need close investigation to see the reasons "behind the scene". However, the underlying data models vary over different datasets, which makes it non-trivial to define outliers in a universal way.

Figure 1.1 shows the ambiguity and difficulties in defining outliers through three basic types of outliers with respect to their data models. Outliers are marked by circles or dotted lines. In this example with only two dimensional data, (a) demonstrates a group of outliers in relation to a normal distribution. The marked outliers can be either extreme values or contaminants emanated from another nearby distribution. The data in (b) is relatively structured and can be described by a linear regression model. We can see that none of the three marked outliers have extreme values in either x or y dimension. They are outliers due to the fact that they deviate from the main body of the data, thus break the expected data pattern. The example indicates that extremes are not necessarily always outliers. Such outliers are indeed very common with multi-dimensional distributions. Figure 1.1(c) is a set of time series data that are prevailing in financial, industrial, meteorological and sociological processes. Outliers in this type of data are presented as those occasional sharp spikes along the time sequence.

Nearly all the datasets are imperfect. They are subject to the contamination of noise, exceptional cases, or incomplete data points, which are part of the overall picture in data collection and recording. Outliers can arise from various mechanisms and occur in any datasets. In statistics literature [4, 6], the causes for outliers are generally divided into two major groups:

- errors in the data that originate from data collection and data entry where inaccuracy or mistakes occur in the process of data reading, calculating or recording; and
- inherent variability of the data, which is common with observational studies where outliers reflects the natural variation of the data;

Depending on the definition of outliers and the specific application domains, it is likely that outliers arise from sources other than the above categories. For example, outliers can be rare events that are surprising or unexpected observations. They have legitimate data values, but do not fit into the main body of the data. For example, most students start university at the age of seventeen or eighteen. However, there are exceptional cases that a few students start university at a much younger age such as fourteen. Such outliers may deliver valuable information about the process under investigation. It is therefore important to identify these outliers so that further study can be conducted to understand what these outliers really represent.

Outlier detection and analysis has long been studied in the literature from statistics community. A common approach in Statistics assumes an underlying distribution model of the data. Statistical discordancy tests are performed to identify some small percentage of points that deviates from the rest of the data. Based mainly on probability distribution and distribution parameters, more than 100 statistical discordancy tests have been developed [6].

With the rapid development of technologies and increased volume of datasets, outlier analysis is now faced with new challenges and requirements. Apart from the size of the datasets, which forms the fundamental difference between classical statistical applications and data mining [22, 32], high dimensionality, diverse characteristics and constantly evolving are major concerns of the datasets that are commonly used in data mining activities. Statistical modeling methods that require fitting the data points to a stochastic distribution become increasingly difficult and inaccurate to handle complex datasets. Quadratic complexity is hardly acceptable with large datasets. The problem now centers around how to efficiently and effectively identify outliers in very large datasets.

In data mining, outlier detection and analysis is often referred to as outlier mining. The main issues involved in outlier mining research include (1) definition of an outlier for a given dataset; (2) well-performed algorithms; and (3) evaluation methodology, including datasets, visualization and user interaction.

A common definition of outlier mining is as follows: *Given a set of N data objects, find those objects that are far outside the norm in one or more dimensions.* In the case when the expected number of outliers is given, for example, mining the top-n outliers in a dataset [59, 40], outlier mining would be described as: *Given a set of N data objects, and O, the expected number of outliers, find the top* n *objects that are far outside the norm in one or more dimensions.*

Like most of the current work on outlier detection, our work is based on two basic assumptions:

- Outliers are only a very small portion of a dataset. i.e., normal data significantly out-number abnormal data.
- Outliers are inconsistent with and deviate from the rest of the data in one or more dimensions.

Test datasets play an important role in the evaluation of outlier detection. Besides the real-life data collected in numerous situations, some synthetic benchmark datasets would help validate and visualize the results of the designed outlier detection algorithms. Currently, little work has been done on developing benchmark data repositories. Existing outlier detection methods tend to use different data to assess their performance, which may result in biases toward the specific models. One Challenge in data analysis, including data clustering and outlier detection is to build a generic framework and automate the data generation process. Therefore, a data generator that can dynamically produce synthetic datasets with various distributions, shapes and densities is an important task in data mining community.

1.2 Contributions

This thesis addresses the issues in outlier analysis and detection in the area of data mining. There are four major contributions.

- **A new notion of outliers:** We proposed reference-based outliers that are considered in the whole data space as well as each respective dimension with respect to a set of reference points. The new notion integrates the distance-based global outliers and the density-based local outliers. Unlike the traditionally defined outliers that considers the whole dataset from only a single viewpoint, the reference-based outliers are analyzed dynamically in the data space. That is, the degree of density for each data point in a dataset is analyzed from different view points where the reference points are located.

- **An efficient outlier detection approach:** Based on the new notion of reference-based outliers, we proposed a fast and effective outlier detection method. The method is compatible with distance-based outlier approaches yet capable of identifying local outliers in terms of data patterns in a dataset. The execution time of our algorithm is $O(Rn \log n)$ where n is the size of dataset and R is the number of reference points. Candidate outliers are ranked based on the Reference-based Outlier Score (ROS). Experimental results indicate that our method is effective and highly scalable to very large datasets.

- **A generic framework for synthetic data generation:** The system automates the generation of multi-dimensional synthetic data generation. A distribution-based and transformation-based approach is used to systematically generate various datasets in terms of data clusters and outliers. The approach is very efficient and the generated 2D dataset is visualized for user inspection.

- **A GUI-based Web Application for data clustering analysis and outlier detection:** The Web portal serves as a data analysis virtual lab to demonstrate visually how some well-known clustering algorithms works to identify clusters and detect outliers. Registered users are allowed upload datasets and test their datasets with different clustering algorithms running on the server side.

1.3 Organization of this thesis

The rest of this thesis is organized as follows: In Chapter 2, we review the existing outlier analysis methods and algorithms in both statistics and data mining communities. In Chapter 3, we present a new reference-based approach to outlier detection and an efficient algorithm for finding global and local outliers. A synthetic data generation system is discussed in Chapter 4. We conclude the thesis with a brief summary and point to the directions for future work in Chapter 5. Appendix A introduces a web-based data analysis application that serves as an on-line tool to assist the learning of data clustering.

Chapter 2

Related Work

Outlier detection and analysis in the literature are mostly from statistics. In the last decade, it has become an important and active research area in data mining due to the increased demand for finding rare but informative cases in large datasets. Different notions of outliers were proposed and many outlier detection algorithms have been developed. One suggested way to classify the existing outlier detection methods is to divide them into two broad categories: set-based and spatial-set-based as shown in Figure 2.1 [66]. The main idea of such classification is to divide the existing methods into spatial and non spatial groups. We can see that set-based methods consist only of the classical distribution-based approaches, which identify outliers by fitting the data into a stochastic model constructed for one or more attributes. Spatial set-based methods that are further divided into multi-dimensional metric spatial data set and graph-based spatial data set, include almost all the recently developed outlier detection methods. This category of methods considers both attribute values and spatial relationships of the data. One major disadvantage to classify outlier detection in the above way is that different notions of outliers are grouped together, which can cause confusion as to the definition and scope of an outlier, for example, whether an outlier is global or local.

In this thesis, we adopt the classification that is based on the way to define outliers. Under this classification, the outlier detection methods are grouped into five categories, namely statistical methods, including distribution-based and depth-based, clustering-based methods, distance-based methods, density-based methods and machine learning methods. With the goal of mining outliers effectively and

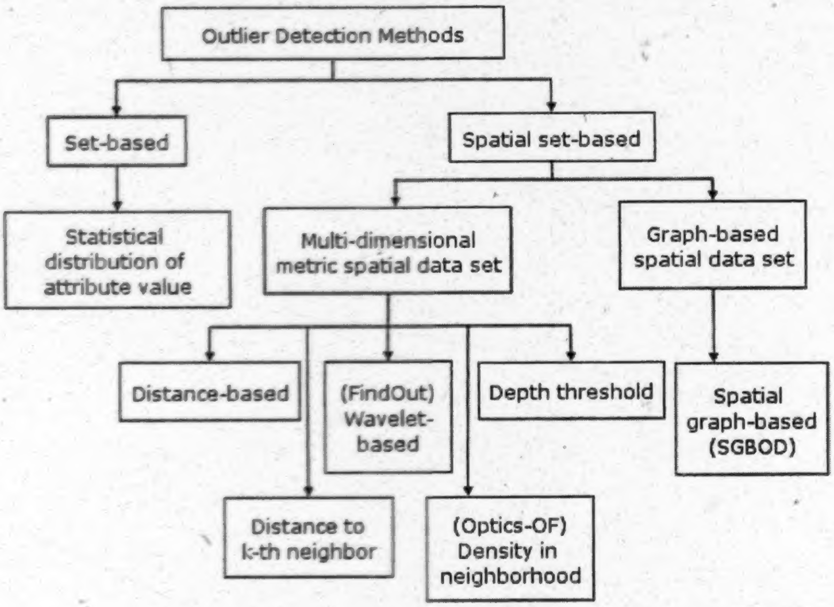

Outlier Detection Methods
Set-based
Spatial set-based
Statistical distribution of attribute value
Multi-dimensional metric spatial data set
Graph-based spatial data set
Distance-based
(FindOut) Wavelet-based
Depth threshold
Spatial graph-based (SGBOD)
Distance to k-th neighbor
(Optics-OF) Density in neighborhood

Figure 2.1: Classification of outlier detection methods in [66]

efficiently, researchers in data mining now focus on developing algorithms that can scale to large and high dimensional datasets. In this chapter, we will concentrate on different notions of an outlier, discuss state-of-the-art algorithms for outlier detection and compare their performance.

2.1 Statistical Outlier Detection

In statistics, the presence of outliers can seriously bias statistical estimates, which results in inference errors and reduction of the power of statistical tests. Therefore, it is critical to identify those extreme or influential data values for reliable and accurate statistical modelling. For decades, many researchers have been engaged in finding ways to detect outliers. Barnett and Lewis discussed approximately a hundred discordancy tests for normal, exponential, poisson and binomial distributions in their book "Outliers in Statistical Data" [6]. Although we do not plan to cover all these discordancy tests, we are going to discuss some of the widely used outlier detection methods in statistics literature.

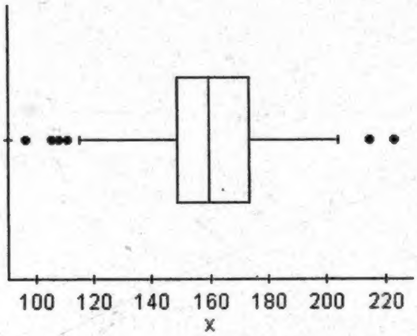


Figure 2.2: Example of a box plot

2.1.1 Graphical tools

There exist quite a few graphical tools to present data for exploratory data analysis and visual inspection, for example, bar graph, pie chart, histogram, line graph and so on. In this section, we will focus on two graphical methods that have been widely used in outlier detection.

Box Plot

A box plot is a graphical approach for conveying the information of data location and variation. It can check outliers in one or more sets of data. Figure 2.2 shows a box plot created using the online software StatCrunch4.0 [68]. In this example, a normally distributed dataset containing three hundred one-dimensional data is examined. The box plot provides us with a five marked summary of the important features of the data distribution. The line in the center of the rectangular box at 160 represents the median, which divides the data into two equal halves. The two ends of the rectangle at 149 and 174, represent the lower quartile and the upper quartile, of which the lower quartile is defined as the 25th percentile and the upper as the 75th percentile. Specifically, 25% percent of the data values are less than 149 while 75% of the data values are less than 174. Thus the box itself has 50% of the data. The whiskers (short vertical line segments at 115 and 214) show the minimum and maximum values of the data being considered normal. They are determined using a

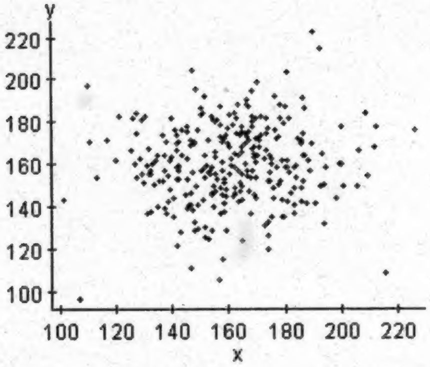


Figure 2.3: Example of a scatter plot

heuristic function that depends on the lower and upper quartiles. Data points outside the whiskers are displayed as outliers. In Figure 2.2, six data points are marked as outliers.

Scatter Plot

A scatter plot is a graphical tool widely used to display measurements of two variables. The resulting pattern indicates the relationship between the two variables. Figure 2.3 is a scatter plot created by StatCrunch4.0 [68]. The dataset in this sample have 300 points in a 2-D space. Both of the variables have a normal distribution and they are generated independently. The plot reveals that variation of y does not depend on x in the dataset. Outliers in the plot are exhibited as the data points lying on the outer skirt of the data body.

In summary, box plots and scatter plots graphically display the spread of the data. They are useful tools for visual inspection of outliers in a dataset. However, the limitation of displaying data with only one or two dimensions has excluded them from receiving much attention from the data mining community.

2.1.2 Distribution-based Outlier Detection

As the name suggests, the definition of a distribution-based outlier is based on a standard probability model, such as Normal, Poisson or Binomial distribution. The most well-known method for detecting distribution-based outliers is to compute

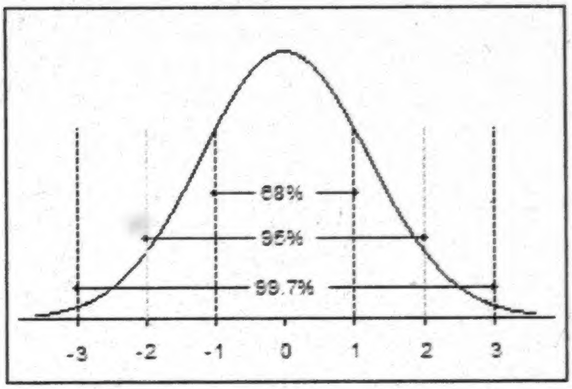


Figure 2.4: standard normal distribution

z-score for each data point. In a z-score test, the data is fitted into a normal distribution. The z-score for each data point indicates how far and in what direction it deviates from the mean of the data distribution, in other words, it is the number of standard deviations from the mean. The value of z-score is calculated based on the mean μ and standard deviation σ of the entire dataset. The formula for z-score is as follows:

$$z = \frac{x - \mu}{\sigma}.$$

Outliers are identified by following the heuristic that any data point with a z-score greater than three is an outlier. Figure 2.4 illustrates a standard normal distribution where the mean is 0 and the standard deviation is 1. As shown with vertical lines of different shades of the color, about 68%, 95%, and 99.7% of the data fall within 1, 2, and 3 standard deviations from the mean in a normal distribution, which explains why $z = 3$ is usually used as a cutoff value to differentiate between normal and abnormal. While the rule is simple as well as effective in many situations, this method suffers from inaccuracy when the data is skewed, especially when the sample size is small since the test parameters mean and standard deviation are obtained from the entire data, including the outliers.

For the multivariate cases, a common method that has been used in statistics is the Mahalanobis distance which, roughly speaking, is defined as the distance between two data points weighted by the covariance matrix of an assumed underlying distribution.

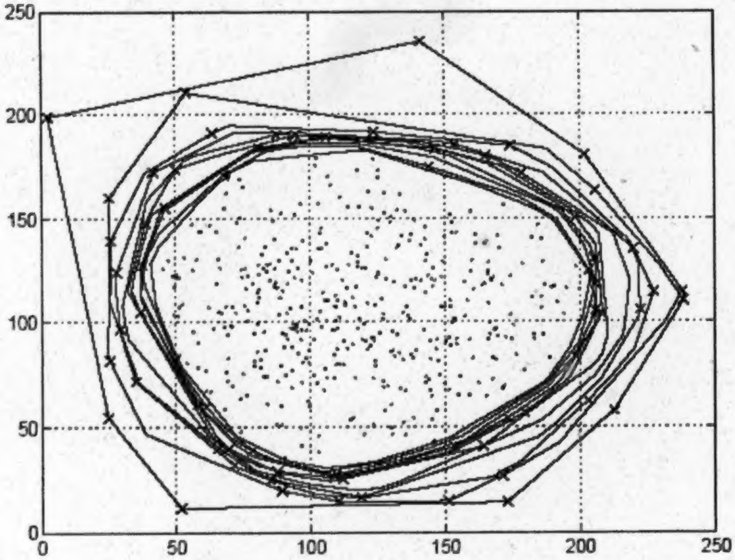


Figure 2.5: Some depth contours for a dataset containing 500 points

While a large number of discordancy tests have been developed for different scenarios, there are two key drawbacks of distribution-based techniques. The first drawback is obvious, i.e., they require *prior* knowledge of the data distribution. Another drawback is that most of the statistical tests are designed in terms of a univariate distribution and they are specific to certain distributions and number of outliers. In numerous data mining activities, we do not usually have enough knowledge about the underlying data model or expected number of outliers. Moreover, fitting a set of complex data into a standard distribution is too expensive and in some cases, impossible. The ever-increasing data size and attributes are also negative factors that prevent distribution-based methods from being applied to practical problems in data mining activities.

2.1.3 Depth-based Outlier Detection

Based on computational geometry, depth-based method organizes data in layers of a convex hull. The method then computes the depth contours [63] or different layers of the convex hull. Outliers are expected to be found in the data lying in the outside perimeter layers. To be more concrete, we borrow the graph from [44] showing the first few depth contours for a dataset having 500 points in Figure 2.5.

The advantage of depth-based methods is two-fold. On one hand, it avoids the assumption of a probability distribution for the data. On the other hand, it is able to identify outliers without requiring any distance function in the feature space which is an advantage over some newly developed methods such as the distance-based methods to be discussed in the next section. However, the key issue with depth-based method is that the complexity of computation of a convex hull with k-dimension has a lower bound $\Omega(N^{\frac{k}{2}})$, which makes this approach not very useful for large high dimensional datasets.

2.2 Outlier Detection in Clustering Analysis

Being one of the unsupervised learning tasks, clustering is the process of grouping similar data objects into classes. Research on clustering analysis has attracted attentions from different areas, including statistics, data mining, machine learning, text retrieval and document categorization. While there has been a rich literature on clustering analysis [31], recent efforts have been focused on finding efficient and effective clustering methods that can handle large multi-dimensional datasets.

Existing clustering methods can be broadly classified into four categories: partitioning methods, hierarchical methods, density-based methods and grid-based methods. While all these methods aim to find meaningful clusters in the database, some have been designed to be able to detect outliers while the clustering is conducted. We will briefly discuss the ideas used in some of the clustering algorithms where outlier detection is considered.

CLARANS [53], a partitioning clustering method gives each object a *silhouette coefficient* to specify how much the object belongs to a cluster. An outlier is identified if the silhouette coefficient is below a certain threshold. The hierarchical method BIRCH [53], marks an object as an outlier if it is far enough from the representative object, or seed. WaveCluster [65] is a grid-based method that is able to filter out noise or outliers through Wavelet transforms. In all of the above methods, outliers are merely identified as a by-product of the clustering process.

The density-based method DBSCAN [20] defines outliers among the data objects. Outliers have a low density in their neighborhood so that they can not be grouped into any clusters. A more recent work [17] shows some interesting relations between DBSCAN and some outlier detection methods to be discussed in the next section. It demonstrates that DBSCAN and one of the distance-based approaches, DB-outlier [1], are almost complementary; and DBSCAN is also complementary with density-based and connectivity-based outlier schemes within a density cluster or far away from some clusters. Interested readers are referred to the work in [17] for details. TURN* [24], a density-based approach developed in our research lab, clusters data objects based on a series of resolutions. A density factor is defined for each object and outliers are those external points with low density values.

Clustering methods aim to achieve optimization in classifying data into groups. Outliers are considered noise and once identified, are usually removed to reduce their influence on the clustering process. The major issue of using these methods to detect outliers is the efficiency and accuracy. Since normal data accounts for the majority part of a dataset, the computation for clustering data objects is often expensive, which in most cases is unnecessary for outlier detection. A well-designed outlier detection method should be able to find meaningful outliers without worrying about how data are assigned to clusters.

2.3 Distance-based Outlier Detection

2.3.1 Basic Concepts

An important topic in outlier detection and analysis is the formulation, or formal definition of outliers. As discussed above, most statistical approaches model the data points using a probability distribution such that an outlier is defined with respect to the underlying data model. Such definition is not feasible for most data mining activities due to the increasing difficulty and cost of fitting large and multivariant data to stochastic models.

---

[1] An object O in a dataset T is a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O [45].

The concept of distance-based outlier was introduced in 1998 by Knorr and Ng [45]. A distance function is used as a metric to identify outliers in large datasets with unknown probability distribution. The basic idea is intuitive and easy to understand, i.e., points that do not have enough neighbors are outliers. Therefore, outliers are identified on the basis of the nearest neighborhood density. Thus, how to define the local neighborhood of a point is the essence of the distance-based method.

Defintion 2.3.1. *An object O in a dataset T is a $DB(p, D)$-outlier if at least fraction p of the objects in T lies greater than distance D from O [45, 47].*

An object is marked as a $DB(p, D)$-outlier if less than or equal to $(1 - p) * N$ objects are within distance D of the object. In other words, a point is considered normal if it has sufficient close neighbors. If the neighbors of a point lie relatively far away, it is regarded as exceptional and classified into outliers. Figure 2.6 shows a small dataset containing 105 points. To identify $DB(p, D)$-outliers in this sample dataset, let $p = 95\%$, and $D = 20$. The Euclidean distance is used as the metric between points. We can see that points $o1$ has no neighbors in the circle (radius $D = 20$) and points $o2, o3, o4$ and $o5$ all have less than five points within their respective neighborhood. On the other hand, every point in the big cluster has more than five points within its $D = 20$ neighborhood. Based on Definition 2.3.1, $o1, o2, o3, o4$ and $o5$ are considered $DB(95\%, 20)$ outliers.

Motivated by this work, more research on outlier analysis has been conducted and different definitions of distance-based outliers have been proposed. In [59], the authors argue that the distance parameter D used in $DB(p, D)$-outlier could be hard to figure out *priori* and its selection often involves trial and error. They proposed a method which identifies outliers based on the distance of a point to its k^{th} nearest neighbor. The main idea is straightforward: the farther away a point is from its k^{th} nearest neighbor, the more likely it is an outlier. A ranking mechanism is introduced for outlier identification and the top n points having the lowest rank are declared to be outliers. The definition is formalized as follows [59].

Defintion 2.3.2. *Given an input data set with N points, parameters n and k, a point p is a D_n^k outlier if there are no more than $n - 1$ other points p' such that*

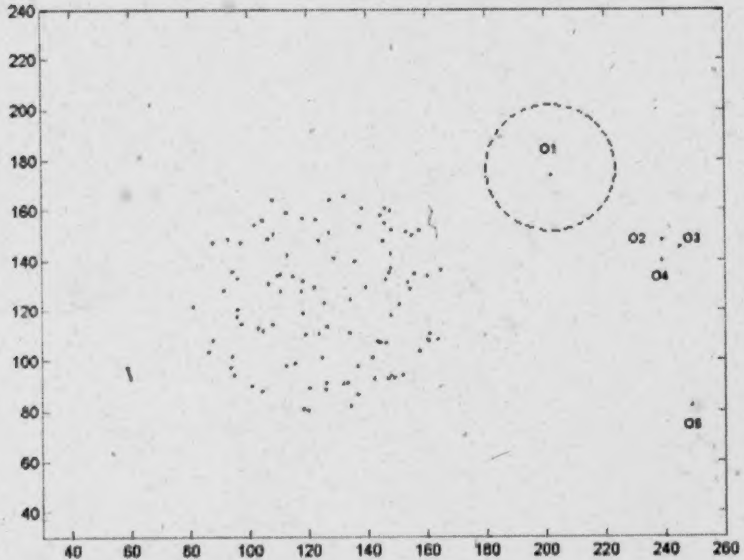


Figure 2.6: Example of $DB(p, D)$-outlier detection, where $p = 95\%$, and $D = 20$

$D^k(p') < D^k(p)$*, where* $D^k(p)$ *denotes the distance of point p from its* k^{th} *nearest neighbor.*

The work enhances distance-based outliers through the use of k^{th} nearest neighbor from the point. The intuition behind this definition is very simple, i.e., given the parameter n which is the number of outliers to be mined, assign each data point a neighborhood density and output the n points that have the lowest density. Therefore, the top n points with the greatest distance to their respective k^{th} nearest neighbors are outliers. With the notion of D_n^k outliers, the neighborhood of a point is specified by its k^{th} nearest neighbor instead of the radius D and density p required in Definition 2.3.1.

Similar to this, Eskin et al [19] reformulated distance-based outliers in terms of the nearest neighbors of a point. The difference lies in that the method used in [19] computes the sum of the distance to the k-nearest neighbors of the point instead of the distance to the k^{th} nearest neighbor. Each data point is assigned a k-NN score, which is the sum of the distance to its k nearest neighbors. A data point is declared as an outlier if its k-NN score is high. The major advantage of this definition over the one in Definition 2.3.2 is that it integrates the information of all the k closest nearest neighbors for a given data point rather than only its k^{th} nearest neighbor.

2.3.2 Algorithms for Mining Distance-based Outliers

Compared to traditional outliers studied in statistics, the definition of distance-based outliers is distribution-free, more flexible, and more computationally feasible. The major issue in detecting distance-based outliers is how to efficiently find such outliers. Since a nearest-neighbor search is required for each of the n data points, straightforward implementations such as the Nested-Loop [45] need to compute the distance between each pair of data points, resulting in an $O(n^2)$ running time. In an effort to quickly find outliers in a large dataset, a number of researchers have been working on improving the efficiency of the algorithm by eliminating unnecessary computation. Techniques used for this purpose include different index structures, pruning rules and the partition of the feature space. The common goal is to develop such algorithms that are scalable to large and high dimensional datasets.

In the following discussion of distance-based algorithms, N is the number of data objects in a dataset, and k is the dimensionality of the data.

Index-based Algorithms

Since distance-based methods define an outlier in term of its neighborhood, i.e., the distance to its close neighbors, finding distance-based outliers can be transformed to the problem of nearest neighbor search for each point. A spatial indexing structure such as R-trees [30], KD-trees [9, 64] or X-trees [10] can be used to organize the data so that the search for neighbors of a query point can be sped up. Outliers are selected among those candidates that have least number of close neighbors or farthest from its neighbors depending on the definition.

Using a spatial index, the number of distance computation can be significantly reduced. Analysis of the approach reveals that it works extremely well and the average running time can be reduced to $O(n \log n)$. Since most of the index structures are designed to work well only in low to moderate dimensional spaces, as dimensionality increases, nearest neighbor search with index structure gets progressively harder. Under the worst case scenario, the complexity of an index-based algorithm is $O(kn^2)$, where k is the number of dimensions.

Nested-loop (NL) Algorithms

A well-known class of methods in outlier detection is the nested-loop algorithm in which distance between points is calculated iteratively. The algorithm is straightforward but the obvious drawback is its quadratic complexity due to the pairwise distance computations between data points. A number of methods have been proposed to improve its efficiency. In order to avoid brute-force search, Knorr and Ng [45] proposed the block-oriented NL algorithm that reads data in blocks so that IO access is minimized.

Recent study on nested-loop algorithms for finding outliers has achieved a near linear time performance given that the input data is in random order [8], which can be done in linear time by repeatedly shuffling the data into random piles and then putting these piles together. The key point in the modified NL algorithm is the use of a pruning rule so that a data point is removed as soon as it is classified as a non-outlier. The number of distance computation is therefore significantly reduced. Table 2.1 shows the details of this variation of the nested loop algorithm presented in [8].

The algorithm assumes that the input data are already in random order. The score function is based on the distance from a point to its k nearest neighbors. The algorithm proceeds by keeping track of the closest neighbors for each data point. The score, which is either the distance to the kth nearest neighbor, or the average distance to the k neighbors, of the weakest outlier found so far is set as the cutoff value that increases while more data are processed leading to more outliers being found. Data points that have a score lower than the cutoff value are marked as non-outliers and removed immediately. This reduces the number of distance computation since non-outliers are eliminated at some early stages. With random ordered data points, the algorithm achieves near linear performance and scales well to large multi-variant datasets.

Table 2.1: A variation of NL algorithm for finding distance-based outliers [8]

```
Procedure: Find Outliers
Input: $k$, the number of nearest neighbors; $n$, the number of outliers to return;
$D$, a set of examples in random order.
Output: $O$, a set of outliers.
Let maxdist($x, Y$) return the maximum distance between $x$ and an example in Y.
Let Closest($x, Y, k$) return the $k$ closest examples in $Y$ to $x$.
begin
    $c \leftarrow 0$          //set the cutoff for pruning to 0
    $O \leftarrow \phi$          //initialize to the empty set
    while $B \leftarrow$ get-next-block($D$) {          load a block of examples from $D$
        Neighbors($b$) $\leftarrow \phi$ for all $b$ in $B$
        for each $d$ in $D$ {
            for each $b$ in $B$, $b \neq d$ {
                if | Neighbors($b$) | < $k$ or distance($b, d$) < maxdist($b$, Neighbors($b$)) {
                    Neighbors($b$) $\leftarrow$ Closest($b$, Neighbors($b$)$\cup$ $d$, $k$)
                    if score(Neighbors($b$), $b$) < $c$ {
                        remove $b$ from $B$
                    }}}}
        $O \leftarrow$ Top($B \cup O, n$)          // keep only the top $n$ outliers
        $c \leftarrow$ min(score($o$)) for all $o$ in $O$     //cutoff = score of the weakest outlier
    }
    return $O$
end
```

Partition-based Algorithm

A few partition-based algorithms have been developed to speed up the search for outliers. By partitioning the space into regions, such algorithms aim to prune out data points that cannot be outliers. Distance computation for finding outliers is conducted only among the remaining data, which can be far less than the original data.

One of such methods is proposed by Knorr and Ng [45], in which the data space is partitioned into equal sized cells. After each data point is mapped to an appropriate cell, the algorithm quickly eliminate two types of cells: (1) cells that contain too many data points, and (2) cells that are the immediate neighbors of those in (1). Execution time is thus significantly reduced since a large number of data points are pruned without implicit distance computation. As pointed out in [45], cell-based algorithm has a time complexity linear in the data size, but exponential in the dimensionality k when $k > 2$. Experiment results reveal that NL algorithm outperforms cell-based algorithm when $k >= 5$.

Another version of partition-based algorithm uses a linear time clustering algorithm such as the pre-clustering phase of BIRCH [79] to generate the desirable partitions by clustering the dataset [59, 19]. In [59], the authors show that partition-based methods outperform both the index-based and block NL algorithms.

Projection-based Algorithms for High Dimensional Data

Most of the outlier methods we have discussed compute the distance between points in the full feature space. Such distance measure can be meaningless for many applications where data contain hundreds of dimensions because data are sparsely distributed in high dimensional space. The notion of finding meaningful outliers in very high dimensional data becomes far more complex. To tackle this problem, a few solutions have been proposed. Aggarwal and Yu in [2] addressed this problem by defining outliers in lower dimensional projections. Outliers are identified according to the density distribution of data projections onto subspaces. Before projection is performed, data are first discretized into grids and each attribute of the data is divided equi-depth ranges. A sparsity coefficient is used to measure the density of the projected data cube. Cubes having negative sparsity coefficient contain data with low densities. Two algorithms were discussed in [2]: a naïve brute force algorithm and an evolutionary algorithm. It is shown that the brute-force algorithm is slow but gives the best results. The fundamental problem lies in that it can hardly deal with high dimensional data. Compared to the brute-force methods, the evolutionary algorithm is close in effectiveness in most cases but far more efficient.

2.3.3 Summary of Distance-based Outlier Detection Methods

The notion of $DB(p, D)$-outliers is a breakthrough in outlier detection in data mining. It generalizes the notion of distribution-based outliers [45] and is more suitable for numerous applications where no standard fitting distribution can be observed for the datasets. Although a reasonable distance function is assumed in $DB(p, D)$-outlier detection, there is no need to define an explicit distribution and the fact that it is computationally feasible for large datasets makes it a promising direction toward outlier analysis.

In summary, distance-based outliers are defined by using some degree of density of a data point relative to its nearest neighbors under a distance metric, or its neighborhood density. Let $X = \{x_1, \cdots, x_n\}$ be the dataset and let $d(\cdot, \cdot)$ be a distance metric. The neighborhood density of a data point $x \in X$ can be defined in the following three ways:

1. 0-1 Density $D(x, k, t)$ where k and t are two parameters: $D(x, k, t) = 0$ if there are fewer than k other data points within distance t and $D(x, k, t) = 1$ otherwise;

2. Max Density $D^m(x, k)$: the reciprocal of the distance to the k-th nearest neighbor;

3. Average Density $D^a(x, k)$: the reciprocal of the average distance to the k nearest neighbors.

It is clear that the first definition divides a dataset into exactly two groups: inliers and outliers. There is no measure of how much a data point is outlying and the

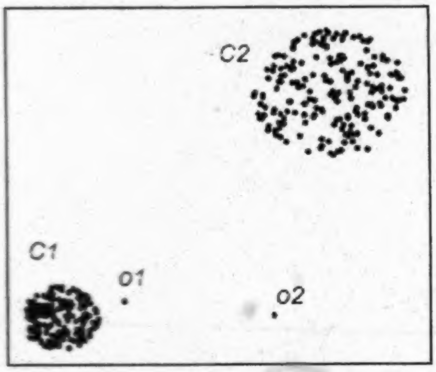


Figure 2.7: Failure of detecting outlier $o1$ using distance-based methods

identified outliers are not ranked. The second and third definitions introduce the ranking mechanism based on the distance to the k nearest neighbors of a point. In such cases, each outlier is assigned a value indicating the degree of its deviation from its close neighbors.

Although distance-based methods enjoy many practical advantages over traditional outlier detection methods, they have their limitations.

- A distance-based method tends to find outliers global to the whole dataset, which is adequate for homogeneous data or data with consistent characteristics, but is not satisfiable for datasets consisting of clusters of diverse density. A typical example is shown in Figure 2.7.
- Only binary property is assumed for each data point, i.e., either an outliers or not.
- Outliers are either not ranked or ranked based purely on the distance to a point's k nearest neighbors. While the ranking is appropriate for global outliers, it can be misleading for outliers local to their own neighborhood.

In Figure 2.7, the dataset contains two clusters $C1$ and $C2$ as well as two outliers $o1$ and $o2$. With the appropriate selection of parameters(p and D or a single k), $o2$ can be easily marked as an outlier by using distance-based methods. In contrast, $o1$ can not be successfully marked as an outlier since it has similar neighborhood density as many points in $C2$. In such case, it is almost impossible to determine the right values for the parameters. If the distance parameter D as defined in Definition 2.3.1 has a big value, $o1$ is very likely to be grouped into cluster C1. Otherwise, it will be an outlier along with many of the points in cluster $C2$. The problem of being unable to identify outliers local to their own neighborhood leads to the discussion in the next section on the local outlier detection.

2.4 Local Outlier Detection

In an effort to overcome the shortcomings of distance-based methods and effectively discover meaningful outliers for a wide variety of datasets, Breunig et al. [12, 13] advocated finding density-based local outliers. A new concept *local outlier factor (LOF)* was introduced to measure the degree to which an object is isolated from its surrounding neighborhood. Another work is the connectivity-based method, which is developed to enhance the LOF scheme by effectively identifying outliers that cannot be distinguished from the surrounding data patterns exclusively by neighborhood density.

In the following subsections, we will review the definition of local outlier and the algorithms to mine local outliers.

2.4.1 Density-based Outlier Detection: Local Outlier Factor (LOF)

Unlike the definition of distance-based outliers that is intuitive and self-explainable, the definition of LOF is based on a number of new concepts as defined in [13].

Let D be a dataset. Let k be a positive integer and let $d(p, q)$ denote the distance between two objects p and q in D.

Defintion 2.4.1. (k-distance of p)
The k-distance of data point p, denoted as k-distance(p), is defined as the distance $d(p, o)$ *between p and data point o such that:*

1. *for at least k objects* $o' \in D \setminus \{p\}$ *it holds that* $d(p, o') \leq d(p, o)$, *and*
2. *for at most* $k - 1$ *objects* $o' \in D \setminus \{p\}$ *it holds that* $d(p, o') < d(p, o)$.

The uniqueness of the above definition lies in that k-distance of p is defined with respect to a specific data point o. With the traditional k^{th} nearest neighbor definition, if there are more than one k^{th} nearest neighbors, only one of them is used in density calculation. With this definition, all of these k^{th} nearest neighbors are used in density calculation.

Defintion 2.4.2. (k-distance neighborhood of p)

The k-distance neighborhood of p is a subset of data points that contains every object whose distance from p is not greater than the k-distance. It is denoted as

$$N_{k-distance(p)}(p) = \{q \in D \setminus \{p\} | d(p,q) \leq k - distance(p)\}.$$

As the object o defined in k-distance(p) may not be unique, it is possible that there are more than k objects in the k-distance neighborhood of p. Though we call $N_{k-distance(p)}(p)$ the k-nearest neighbors of p, the number of items in the nearest neighbor set is actually greater than or equal to k.

Defintion 2.4.3. (reachability distance of p w.r.t object o)

The reachability distance of object p with respect to object o is defined as

$$reach - dist_k(p,o) = max\{k - distance(o), d(p,o)\}.$$

Note that the higher the value of k, the more likely the reachability distance for objects within the same neighborhood has the same value.

Defintion 2.4.4. (local reachability density of p)

The local reachability density of p is defined as

$$lrd_k(p) = 1/[\frac{\sum_{o \in N_k(p)} reach - dist_k(p,o)}{|N_k(p)|}].$$

Clearly, the local reachability density for an object p is the reciprocal of the average distance between p and those objects in its k-neighborhood. It is an estimation of the density around p's neighborhood.

Defintion 2.4.5. (outlier factor of p)

The local outlier factor of p is defined as

$$LOF_k(p) = \frac{\sum_{o \in N_k(p)} \frac{lrd_k(o)}{lrd_k(p)}}{|N_k(p)|}.$$

LOF is the average of the ratio of the local reachability density of p and those of p's k-nearest neighbors. What distinguishes it from the previous outlier detection methods is that it takes into account not only the neighborhood density of p but also the neighborhood densities of its k- nearest neighbors. Being assigned a LOF value, each data object has a degree of being outlying relative to its local cluster, i.e., the cluster it is in, or the clusters that are close to it. A higher value of LOF for an object p indicates that p lies in a sparse region in the context of its local neighborhood. Some general properties of LOF include

- objects deep in a cluster have an LOF value close to 1, which favors uniformly distributed data;
- LOF value changes non-monotonically with k.
- Generally, LOF(p) is bounded and the lower and upper bounds are associated with p's direct and indirect neighborhood.

Algorithms for Finding Density-based Local Outlier Factor

A two-step algorithm has been proposed to detect local outliers. In the first step, the algorithm finds the k-nearest neighbors for each point. A spatial index structure has been used to improve the efficiency. The average complexity for k nearest-neighbor query is $O(n \log(n))$ with low dimensional data. In the second step, the local outlier factor (LOF) is computed. The algorithm scans the database twice. The first scan finds the estimation of the density for each object so that LOF can be calculated in the second scan. The time complexity of this step is $O(n)$.

Experiments show that density-based method is more powerful in identifying local outliers than distance-based methods. However, the tradeoff is the high computation cost. In order to improve the efficiency, Jin et al. introduced in [40] the concept of "micro-cluster" and proposed the micro-cluster-based local outlier detection algorithm. Observing that outliers are only a very small portion of a dataset, Jin et al. attempt to reduce the computation cost by constraining the k-nearest-neighbor search to only the top-n local outliers. A "cut-plane" method is designed to identify the boundary between a data point and a constructed micro-cluster. The algorithm works in three steps.

- Building micro-clusters using BIRCH's [79] preclustering algorithm.
- Computing upper and lower bounds on the LOF value for micro-clusters.
- Detecting top-n outliers by their rank.

Experiments conducted in [40] indicate that the micro-cluster TOP-n LOF mining method outperforms the two-step algorithm. It is scalable to large and high dimensional datasets.

2.4.2 Connectivity-based Outlier Detection: Connectivity-based Outlier Factor(COF)

As shown in Figure 1.1 (b) in the introduction, an important type of outliers is the structured outliers that are considered w.r.t. data pattens. They are data points that do not fit into the data model or data pattern. Such outliers may or may not have similar densities as the data patterns in their neighborhood. Although LOF suffices for identifying outliers having lower density than their neighborhood density, it may rule out outliers that have similar density with the neighboring non-outlier patterns. The connective-based outlier detection scheme was proprosed by Tang et al. in [71]. The motivation behind this scheme is to separate the notion of low density from that of isolativity. The notion of "low density" refers to the fact that an object is in a relatively sparse region while "isolativity" refers to the degree that an object is connected to other objects. With this scheme, each data object is provided with a connectivity-based outlier factor (COF) measuring the degree to which it is deviating from a connected pattern. Figure 2.8 is an instance of an isolated outlier used in [71]. For a dataset like this, LOF would fail to identify the single outlier $o1$ due to the low density patterns around its neighborhood.

While we do not provide the detailed concepts and definitions for COF, we will illustrate the method using a simplified example as shown in Figure 2.9. For formal definitions, please refer to [71] for details.

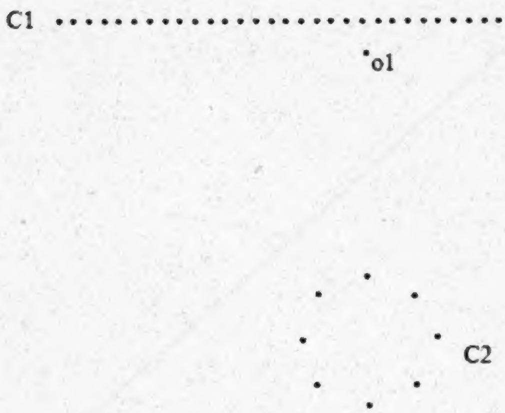
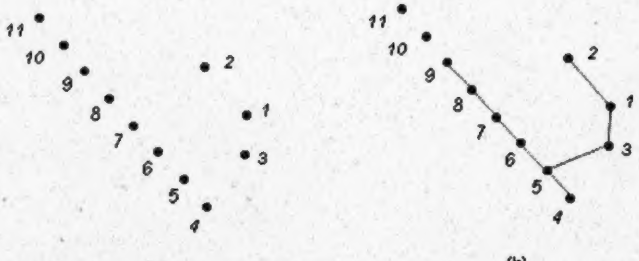


Figure 2.8: Failure of Outliers detection using LOF


11
10
9
8
7
6
5
4
2
1
3
11
10
9
8
7
6
5
4
2
1
3

(a) (b)

Figure 2.9: A simplified example showing outlier detection using COF

Let D be a dataset. Let p be an object in D. Let k be a positive integer. Let $d(p, q)$ denote the distance between p and q. Let $G = \{p_1, p_2, ..., p_r\}$, where r is a positive integer, be a subset of D. Let $N_k(p)$ be the k-neighborhood of p as defined for LOF. In the example in Figure 2.9, $D = \{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$, $k = 8$.

SBN-path (set based nearest path)

The SBN-path of an object expands itself by including the nearest neighbors. As a result, SBN-path exhibits the order in which the nearest objects are presented. For neighbors having the same distance to an object, a pre-defined order is taken so that the SBN-path is unique for each object.

In Figure 2.9 (a), the k-neighborhood of object 1 is $N_8(1) = \{3, 2, 5, 6, 4, 7, 8, 9\}$, thus,

$$s_1 =< 1, 3, 5, 4, 6, 7, 8, 9, 2 > .$$

SBN-trail (set based nearest trail)

From Figure 2.9 (b), we have SBN-trail for data point 1 as

$$tr_1 =< (1, 3), (3, 5), (5, 4), (5, 6), (6, 7), (7, 8), (8, 9), (1, 2) > .$$

Average chaining distance

The average chaining distance from p_1 to $G \setminus p_1$ is defined as

$$ac - dist_G(p_1) = \frac{1}{r-1} \sum_{i=1}^{r-1} \frac{2(r-i)}{r} \cdot dist(e_i).$$

The average chaining distance is the average of the weighted distances in the cost description of the SBN-trail. It provides a measure of how tight the objects on an SBN-trail are chained. It is the use of *weight* here that differentiating COF from LOF. Corresponding to the average chaining distance, the local reachability density of an object p calculated for LOF treats all the objects in p's k distance neighborhood in the same manner. As a consequence, LOF can not successfully detect certain outliers if an improper value of k is used. By adding weight in the computation of average chaining distance, COF distinguishes the roles of the neighboring points.

Larger weights are assigned to the earlier items appeared in SBN-trail. Therefore, edges close to p_1 contribute more for $ac - dist_B(p_i)$ if they have large values.

Consider the dataset in Figure 2.9. Suppose we have the following cost description (length of edges) for object 1:

$$c_1 = \{1, 2, 1, 1, 1, 1, 1, 3\},$$

we can compute the average chaining distance for object 1 and have

$$ac\text{-}dist_{N_k(1)\cup\{1\}}(1) = \frac{2}{(9-1)\cdot 9}(8\cdot1+7\cdot2+6\cdot1+5\cdot1+4\cdot1+3\cdot1+2\cdot1+1\cdot3) = 1.25$$

COF (connectivity-based outlier factor)

The connectivity-based outlier factor at object p with respect to its k-neighborhood is defined as

$$COF(p) = \frac{|N_k(p)| \cdot ac - dist_{N_k(p)}}{\sum_{o\in N_k(o)} ac - dist_{N_k(o)}}$$

COF of an object p is the ratio of the average chaining distance of p to the average of the average chaining distance of p's k-distance neighbors. It captures the degree to which an object is shifted away from the surrounding pattern or structure. High value of COF for an object indicates that the object is strongly shifted away from its close pattern and is more likely to be an outlier.

Apart from the capability to identify outliers deviating from low density patterns as demonstrated with the above example, it is indicated in [71] that connective-based method is nearly as powerful as density-based method in detecting outliers deviating from high density patterns. Like LOF, COF also has the two properties.

- COF value for an object deep inside a cluster is close to 1.
- There exist upper and lower bounds for COF.

2.4.3 Algorithms for Finding Connectivity-based Local Outlier Factor

The design of the COF algorithm is identical to that of the two-step algorithm for obtaining LOF. In the first step, the algorithm searches for the k-nearest neighbors and computes the SBN-trails for each object. COF values are calculated in the second step. By applying an index tree, this algorithm has a similar time complexity as the LOF algorithm.

2.4.4 Summary of Local Outlier Detection

In general, the notion of an outlier based on a comprehensive outlier factor is very important in outlier detection for real world datasets. In many applications, datasets are often more complex and can exhibit various characteristics. The definition of outlier factor (LOF or COF) is able to capture the degree to which an object is deviating from the other points in its neighborhood. One remarkable common property of such score-based methods is that the identified local outliers are ranked, which makes it superior in dealing with datasets containing different clusters and patterns (in density and shape) than many existing methods.

Both LOF and COF require a user-defined parameter k, or $MinPts$. The authors in [13] presented some guidelines for choosing k within a range of values. However, it remains to be an issue as to how to select an appropriate value for a user-defined parameter in any outlier detection method.

2.5 Neural network Outlier Detection

Neural network has proven to be an effective approach to classifying patterns. A Replicator Neural Network (RNN) is a feed-forward multi-layer perceptron network with three hidden layers. This type of neural networks has powerful approximation capabilities. They have been used as a tool in image and speech processing [1, 35]. RNN-based outlier detection method was first proposed by Hawkins et al. in [34]. Figure 2.10 is a schematic view of the fully connected Replicator Neural Network used in [34].

An RNN model is trained from a set of sample data by capturing the most important features. The input variables are also the output variables so that an implicit and compressed data model is constructed during the training phase. Reconstruction error is then used as the measure indicating how far an individual object is

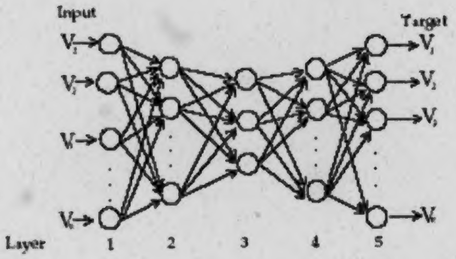


Figure 2.10: A schematic view of a fully connected replicator neural network

outlying. The insight is that outliers tend to be reproduced poorly by a trained data model. *Outlier Factor (OF)* is defined as the average reconstruction error over all features for each data object. It provides a score for measuring how much a certain object is outlying against the rest of the data in the dataset. Since outliers deviate from the common data pattern, they tend to have higher reconstruction errors, leading to a higher value of OF. The RNN-based method consists of three steps. First, sampling the data; second, training of the RNN; and third, Computing OF value for each data object. Outliers are ranked according to their OF scores. The higher the score, the stronger an outlier is.

A comparative study conducted between RNN and three other methods is presented in [76] by the same authors who developed the RNN method. They provide an empirical evaluation of the RNN approach based on both small statistical datasets and large data mining datasets. The experiment results show that RNN method performs satisfactorily for both small and large datasets. However, the drawback of long training cycles which leads to high computation costs often prevents RNN-based method from being applied to practical problems. In addition, like other flexible nonlinear estimation methods (e.g., kernel regression), RNN-based method can suffer from either underfitting or overfitting the data, which will affect the effectiveness of outlier identification.

Chapter 3

An Efficient Reference-based Approach to Outlier Detection in Large Datasets

A bottleneck to detecting distance and density based outliers is that a nearest-neighbor search is required for each of the n data points, resulting in a quadratic number of pairwise distance evaluations. In this chapter, we propose a new method that uses the relative degree of density with respect to a set of reference points to estimate the neighborhood density of a data point. The running time of our algorithm based on this approximation is $O(Rn \log n)$ where n is the size of the dataset and R is the number of reference points. Candidate outliers are ranked based on the outlier score assigned to each data point. Theoretical analysis and empirical studies show that our method is effective, efficient, and highly scalable to very large datasets.

3.1 Motivation

Detecting distance-based outliers has attracted much attention over the last decade. Compared to traditional outliers studied in statistics [6, 33], the definition of distance-based outliers is distribution-free, more flexible, and more computationally feasible. A bottleneck to the detection of distance-based outliers is that a nearest-neighbor search is required for each of the n data points. Consequently, straightforward implementations such as the Nest-Loop method need to compute the distance between each pair of data points, resulting in an $O(n^2)$ running time.

Since the seminal work of Knorr and Ng [45], much effort has been devoted to improving the efficiency of algorithms for detecting distance-based outliers. By using spatial index data structures such as the k-d tree and its variants, the average running time can be reduced to $O(n \log n)$ with a hidden constant depending exponentially on the dimension of the data. Several heuristics have also been proposed to reduce the number of required nearest neighborhood search. In [45], a cell-based approach for detecting distance-based outliers was investigated, which is still exponential in the dimensions, but linear in the size of the dataset under the assumption that both of the two negatively-correlated parameters (percentage p and radius D) of the algorithm are set to their ideal values. In [59], some clusters of data points are eliminated from consideration based on the result of pre-clustering the dataset, a task that is dual to outlier detection. In [8], it is observed that by keeping track of the closest neighbors found so far, the nearest neighbor search for a specific data point can be stopped if it becomes clear that the data point cannot be one of the pre-specified number of outliers. While the algorithm can indeed prune many distance calculations, the worst-case running time is still $O(n^2)$. Empirical evidence and theoretical arguments under some assumptions in [8] show that the algorithm based on this observation may have a sub-quadratic running time in practice. The study of fast algorithms for the nearest neighbor problem is a traditional topic in algorithm and there is a huge amount of literature discussing various approaches to solve the nearest search problem in a time and space efficient way. For the case of two dimensional space, there exist algorithms that solve the problem in $O(n \log n)$ time. For datasets in high dimensional space, there are a variety of exact, approximation, and randomized algorithms with different time and space time complexity [26, 39].

In this chapter, we propose a new approach to reducing the number of distance evaluations. The idea is to rank the data points based on their relative degree of density with respect to a given set of reference points. For each reference point, we calculate its distance to each of the data points and transform the original data space into a one dimensional dataset. Based on the obtained one-dimensional dataset that contains the distances from a reference point to each data point, the relative degree of density (w.r.t the reference point) of each data point is calculated. The overall relative degree of density of a data point is defined as the minimum relative degree of density over all the reference points. The running time of the algorithm is $O(Rn \log n)$ where R is the number of reference points and n is the size of the dataset. The method is further optimized by gradually increasing the number of reference points. For distance measures that satisfy the triangle inequality, data points identified by our reference-based method as outliers are most likely considered as outliers by the distance-based approach. In addition to the properties that distance-based method has, our approach can also find local outliers specific to various data patterns in complex datasets.

In the following discussion, let $X = \{x_1, \cdots, x_n\}$ be the dataset and let $d(\cdot, \cdot)$ be a distance metric.

3.2 Review of Distance-based Outliers

Distance-based outliers are defined by using some degree of density relative to the nearest neighbors, or the so called neighborhood density of a data point under a distance metric [45, 59, 19]. Let x be a data point in X, k be an integer and t be a real number, the neighborhood density of a data point $x \in X$ can be defined in the following three ways:

Defintion 3.2.1. *0-1 Density* $D(x, k, t)$*:* $D(x, k, t) = 0$ *if there are fewer than* k *other data points within distance* t *and* $D(x, k, t) = 1$ *otherwise;*

The definition considers being an outlier as a binary property such that the obtained density for each data point divides the whole dataset into exactly two groups: inliers and outliers. There is no measure of how much a data point is outlying and the identified outliers are not ranked.

Defintion 3.2.2. *Max Density* $D^m(x, k)$*:* $D^m(x, k)$ *is the reciprocal of the distance to the* k*-th nearest neighbor.*

Defintion 3.2.3. *Average Density* $D^a(x, k)$*:* $D^a(x, k)$ *is the reciprocal of the average distance to the* k *nearest neighbors.*

Definition 3.2.2 and 3.2.3 introduce the ranking mechanism based on the distance to the k nearest neighbors of a point. The identified outliers are more meaningful since the information of the degree of being an outlier has been integrated into the analysis process.

Some local outlier detection methods [13, 70] generalize the above concepts further. For example, the well-known **local outlier factor** (LOF) introduced in [13] measures the degree of being an outlier by taking into consideration the data point's relative density as compared to those of its nearest neighbors. The advantage of LOF is that the local densities of the non-outlier data points will have less impact on the ranking of the outliers. The major parameter in LOF is *MinPts*, the minimum number of the nearest neighbors to consider. This parameter is highly application-dependent and some insight into the structure of the dataset is required in order to set it correctly. What makes the selection of *MinPts* even harder is the fact that the LOF of a given data point is not monotone in *MinPts*, as has been observed by the authors [13]. Another related issue with LOF is the existence of duplicated data in a dataset. Roughly speaking, the LOF of duplicated data points is infinity unless the *MinPts* is larger than the number of duplicated data points. As has been mentioned in [13], this difficulty can be overcome by slightly changing the original definition of LOF to ignore the neighboring data points that are duplicated.

Our approach follows Definition 3.2.3. Based on the average distance to the k nearest neighbors, each data point is assigned an outlier score indicating the degree of its deviation from its close neighbors. Outliers are those with a low neighborhood density but high outlier scores.

3.3 Reference-based Outlier Detection Method

We use the relative degree of neighborhood density with respect to a given set of *reference points* to approximate the degree of density defined in the distance-based method. Let $X = \{x_1, ..., x_n\}$ be a dataset and p be a point (not necessarily in X). Consider the vector that consists of the distances between p and each of the data

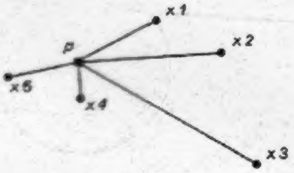


Figure 3.1: Reference-based nearest neighbors in the one-dimensional dataset X^p

points in X:

$$X^p = \{d(x_i, p), 1 \leq i \leq n\},$$

which can be viewed as a one-dimensional representation (w.r.t. p) of the original data.

Defintion 3.3.1. *Given a data point $x \in X$. A data point $y \in X, (y \neq x)$ is a reference-based nearest neighbor of x with respect to the vector X^p if*

$$|d(x,p) - d(y,p)| = \min_{1 \leq i \leq n} |d(x,p) - d(x_i,p)|$$

where the minimum is taken over all the $x_i \in X$ and $x_i \neq x$.

The above idea is illustrated in Figure 3.1, where p is a reference point and the dataset contains only five points: x_1, x_2, x_3, x_4 and x_5. To find the reference-based nearest neighbor of each data point, we first find the distances from each point to p as shown in the figure. We now have a one-dimensional data set, whose values are the distances to p, i.e., $X^p = \{d(x_1,p), d(x_2,p), d(x_3,p), d(x_4,p), d(x_5,p)\}$. For a given data point, the reference-based nearest neighbor is the closest point to it in the one dimensional data space X^p. For example, the reference-based nearest neighbor of x_1 is x_5 and the reference-based nearest neighbor of x_2 is x_1. Intuitively, reference-based nearest neighbors with respect to p are not necessarily the closest in the original dimensional space. For 2D data, points located on the same circle (p is the center) have a reference-based distance of 0. This property indicates that we usually need more than one reference points to improve the distance approximation in order to find global outliers. On the other hand it is a major factor to contribute to the successful identification of local outliers in a complex dataset. We will discuss this in detail in the next section.

Defintion 3.3.2. *Let x be a data point in X and $\{x_1, ...x_k\}$ be the set of k reference-based nearest neighbors to x. The relative degree of density for x in the one-dimensional data space X^p, denoted as $D(x, k, p)$, is defined as*

$$D(x,k,p) = \frac{1}{\frac{1}{k}\sum_{j=1}^{k} \mid d(x_j,p) - d(x,p) \mid}.$$

Given a reference point, the neighborhood density of x is the reciprocal of the average distance to its k reference-based nearest neighbors in the one-dimensional space $X^p = \{d(x_i, p), 1 \leq i \leq n\}$.

Defintion 3.3.3. *Let $P = \{p_1, \cdots, p_R\}$ be a set of R reference points. We define the neighborhood density of a data point x w.r.t. P as*

$$D^P(x,k) = \min_{1 \leq r \leq R} D(x,k,p_r)$$

where k is a fixed parameter, indicating the number of reference-based nearest neighbors considered for each reference point.

Essentially, data points deviating from their surrounding data patterns will get lower neighborhood density values. We will discuss the details in the next few sections. Based on the neighborhood density, each data point is assigned a reference-based outlier score, or ROS, which is defined as

$$ROS(x) = 1 - \frac{D^P(x,k)}{\max_{1 \leq i \leq n} D^P(x_i,k)}$$

Data points in a given dataset are ranked according to their relative degree of density computed on a set of reference points. Outliers are those with higher values of outlier scores.

3.3.1 Compatibility with Distance-based Method

By using a set of reference points, we want to obtain the best approximation of the nearest neighbor search in the original data space, i.e. to approximate the distances between each data point in the original data space. If the distance metric satisfies the triangle inequality, we have the following observations that demonstrate the effectiveness of using the relative degree of density to approximate the traditional definition of density. Consider a reference point p and two data points $x_i, x_j \in X$. Based on the triangle inequality, we have

$$|d(x_i, p) - d(x_j, p)| \leq d(x_i, x_j).$$

Note that the two sides in the above equation are equal when p, x_i and x_j are on the same line.

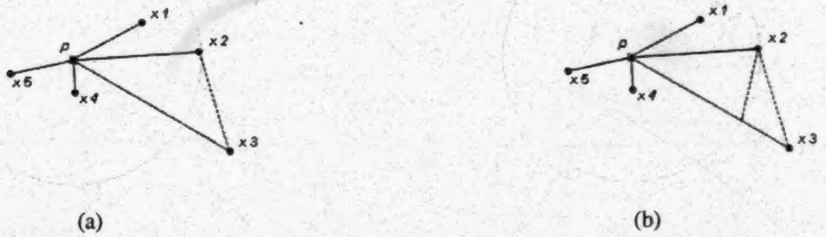


Figure 3.2: Reference-based nearest neighbors satisfying the triangle inequality

Using the dataset in Figure 3.1 as a simplified example, we can see from Figure 3.2(a) that the reference-based nearest neighbor of x_3 is x_2, and

$$d(x_3, p) \leq d(x_2, p) + d(x_2, x_3)$$

if Euclidean distance is used. Therefore, we have

$$|d(x_3, p) - d(x_2, p)| \leq d(x_2, x_3),$$

as outlined in Figure 3.2(b).

Formally, we have the following

Lemma 3.3.4. *For any set of reference points P and any data point $x_i \in X = \{x_1, x_2, \cdots, x_n\}$,*

$$D^P(x_i, k) \geq D^a(x_i, k)$$

where $D^a(x_i, k)$ is the average density as defined in Definition 3.2.3.

Proof. Let $\{x_{i_1}, \cdots, x_{i_k}\}$ be the k nearest neighbors of x_i with respect to the distance in the original data space X. Then,

$$\frac{1}{k}\sum_{j=1}^{k}|d(x_i,p) - d(x_{i_j},p)| \leq \frac{1}{k}\sum_{j=1}^{k} d(x_i, x_{i_j}). \tag{3.1}$$

Although $\{x_{i_1}, \cdots, x_{i_k}\}$ are the real k nearest neighbors defined in the original data space, they are not necessarily the k referenced-based nearest neighbors of x_i in the data space X^p. We show in definition 3.3.1 that reference-based nearest neighbors have the minimum distance difference in X^p. Therefore, the average distance from $\{x_{i_1}, \cdots, x_{i_k}\}$ to p is greater than or equal to that of the reference-based nearest neighbors. Thus we have by Definition 3.3.2

$$D(x_i, k, p) \geq \frac{1}{\frac{1}{k}\sum\limits_{j=1}^{k}|d(x_i,p) - d(x_{i_j},p)|} \tag{3.2}$$

where the left hand side is the inverse of the average distance from $\{x_{i_1}, \cdots, x_{i_k}\}$ to p. From equations 3.1 and 3.2, we get

$$D(x_i, k, p) \geq \frac{1}{\frac{1}{k}\sum\limits_{j=1}^{k} d(x_i, x_{i_j})}$$

Since the above holds for any reference point in P, it follows that

$$D^P(x_i, k) \geq D^a(x_i, k),$$

where $D^a(x_i, k)$ is the average density as defined in Definition 3.2.3. □

The above shows that for a given data point x, the reference-based density is lower bounded by the neighborhood density computed using the traditional k nearest neighbor search method. If a data point has a small enough density to be identified as an outlier using the reference-based approach, it will have an even smaller density value using the distance-based method. If a threshold α is used to determine the outliers, then the above analysis indicates that outliers detected using our reference-based density are also outliers identified by the KNN-based density.

The following special cases are also interesting and provide further justification to our proposal. When the data set is one-dimensional, a single reference point (say 0) is sufficient and our approach is equivalent to the traditional distance-based approach. On the other hand, if the dataset itself is used as the set of reference points, then our approach is identical to the traditional KNN approach. Formally, we have the lemmas which are easy to prove.

Lemma 3.3.5. *Assume that the dataset* $X = \{x_1, \cdots, x_n\}$ *is one dimensional. Then,*

$$D^P(x_i, k) = D^a(x_i, k), \forall x_i \in X$$

for any set P *of reference points.*

Lemma 3.3.6. *For any data point* $x_i \in X$ *that is contained in the set* P *of the reference points, we have*

$$D^P(x_i, k) = D^a(x_i, k)$$

Proof. When x_i is in P, we have

$$D^P(x_i, k) \leq D(x_i, k, x_i) = D^a(x_i, k).$$

The result follows from Lemma 3.3.4. □

3.3.2 Algorithm and Its Implementation

Let $P = \{p_r, 1 \leq r \leq R\}$ be a given set of reference points. The algorithm finds the potential outliers in the dataset X in three major steps:

1. For each reference point $p \in P$, sort the original dataset X in the one-dimensional space $X^p = \{d(x_i, p), 1 \leq i \leq n\}$, i.e., data points in X are ordered according to the distances to p.

2. For each data point $x \in X$, find the k reference-based nearest neighbors and compute the average neighborhood density $D(x, k, p)$;

3. Set $D^P(x, k)$ of each point x to be the minimum of $D(x, k, p_r)$ w.r.t. P and compute the outlier score ROS.

Outliers tend to have a higher value of ROS and they are ranked according to their ROS values. The detail is shown in Algorithm 1.

It takes $O(n)$ time to compute the distance vector X^p for each reference point $p \in P$. The calculation of the average neighborhood density $D(x, k, p)$ involves finding the k reference-based nearest neighbors. Since the reference-based nearest neighbors are calculated on the one-dimensional space X^p, it suffices to find them by sorting the original dataset X using the values in X^p, or the distances to the given reference point p as the key. Sorting the distance vector X^p can be done in $O(n \log n)$ time. Once the distance vector X^p has been sorted, the calculation of $D(x, k, p)$ for each data point $x \in X$ can be done in constant time. Thus, the overall time of Algorithm 1 is $R(n \log n + n)$, which gives a complexity of $O(Rn \log n)$ where R is the number of reference points.

Algorithm 1 The Algorithm for computing $D^P(x, k)$

```
Input: dataset $X = \{x_i, 1 \leq i \leq n\}$, reference point set $P = \{p_r, 1 \leq r \leq R\}$
Let $X^p$ be the vector containing distances to a reference point $p$ and $k$ be a positive integer

$p = p_1$
$X_s$ = mergeSort($X$)   //according to the distance vector $X^{p_1}$
for each $x \in X$ do
    $D(x, k, p)$=computeDensity($x, X_s$)
    $D^P(x, k) = D(x, k, p)$
end for
for each $2 \leq r \leq R$ do
    $p = p_i$   //next reference point closest to $p$
    $X_s$ = sort($X$) according to the distance vector $X^{p_i}$
    for each $x \in X$ do
        $D(x, k, p)$=computeDensity($x, X_s, k$)
        if $D(x, k, p) < D^P(x, k)$ then
            $D^P(x, k) = D(x, k, p)$
        end if
    end for
end for
```

Further Speedup

To further improve the efficiency in computing $D^P(x,k)$, we make the following observation. Assume that p_1 and p_2 are two reference points and that $d(p_1, p_2)$ is small. Then, data points in X sorted according to their distances to p_1 are usually "almost" sorted according to their distances to p_2. Thus, if we have processed p_1 and recorded the corresponding sorted order $X_{p_1} = \{x_{i_1}, \cdots, x_{i_n}\}$, we can calculate $D(x,k,p_2)$ by sorting the ordered list $X_{p_1} = \{x_{i_1}, \cdots, x_{i_n}\}$ with the various adaptive sorting algorithms that can take advantage of the "near sortedness" of the vector X_{p_1}. One example of such adaptive sorting algorithms is the simple insertion sort whose running time is in $O(n + REV)$ where REV is the number of pairs of the elements whose relative order is wrong [21]. Therefore, while the worst case execution time of computing $D^P(x,k)$ is $O(Rn \log n)$, the practical execution time of our algorithm can be much lower.

3.3.3 Determination of Reference Points

In reference-based approach, each reference point is not necessarily a data point in X, it is actually a virtual point. The determination of suitable reference points plays an important role in terms of both the effectiveness and the efficiency of our algorithm.

In our implementation, we use as reference points the vertices on a grid obtained by partitioning the axes in the data space to facilitate the selection of the closest next reference point in the second for loop of the above algorithm. The advantages of using vertices on the grid over randomly selecting reference points is two-fold: (1) Reference points are evenly distributed in the whole data space, and (2) the outlier detection result is deterministic, i.e., the obtained results would be the same with each run. Another way to position the reference points is to put them on a convex hull. While it is good in low dimensional spaces, it is not practical for high dimensional data due to the lower bound complexity of $\Omega(n^{\frac{k}{2}})$ for finding the convex hull, where k is the number of dimensions.

Recall that the overall running time of our algorithm is in $O(Rn \log n)$ where

R is the number of the reference points that is independent of the data size n. It is a constant that is determined by the features of the datasets. For a simple dataset that contains one cluster, a few reference points (say 9 for 2D data) are enough to correctly detect the outliers in even very large datasets. Domain knowledge of a given dataset can usually help in determining the number of reference points. With real world data, we do not usually have such domain knowledge. By using the grid vertices approach, we are able to partition the space incrementally from coarse resolution to fine resolution to determine the appropriate number of reference points. Also, notice that all calculations in the current partition are not lost in the next partition, and only calculation for additional reference points is computed. Due to the sparse property of high dimensional data, it is not necessary to partition the data space based on all dimensions. The grids can be built on just a few dimensions so that the number of reference points remains a constant value. We leave the question of how to select these dimensions for partitioning the space in high dimensional datasets as an open issue for future work.

3.3.4 Detecting Global and Local Outliers in Complex Datasets

The distance-based method is static in that it uses parameters with fixed values for all the data points in a dataset. It ignores the cases where data patterns have different densities, thus considers all data points in a dataset in the same setting..As a consequence, it always assigns low density values to data points located in sparse regions though some of them are deep in sparsely distributed clusters. The authors in [13] also argue that the distance-based outlier detection method can only take a global view of the dataset, resulting in failure to identify outliers local to certain clusters in a complex dataset.

By using a set of reference points, our reference-based approach is dynamic and able to see the whole dataset from various viewpoint. It is possible that the reference-based nearest neighbors of a given data point are different with respect to different reference points. Therefore, at one reference point, the local outliers may have a high neighborhood density due to false nearest neighbors, while at another reference point, it may be shown lying in a very sparse neighborhood. Since

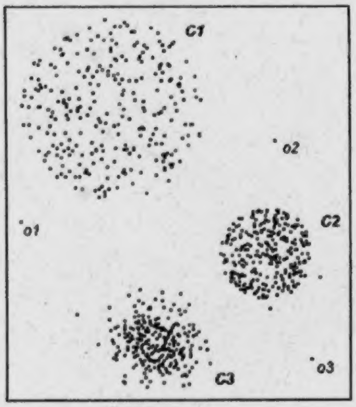
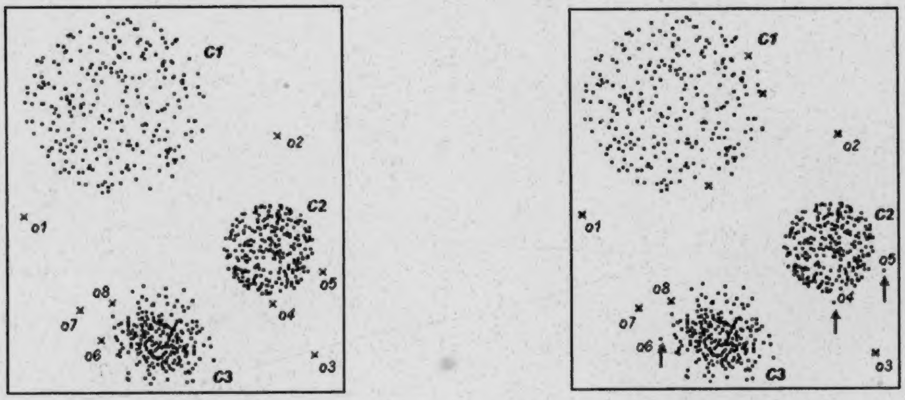


Figure 3.3: A 2D dataset containing multiple clusters with local outliers

the reference-based neighborhood density of a data point x is determined by the minimum among all its densities computed based on the set of reference points, it is guaranteed that with a set of reference points evenly covering the data space, false nearest neighbors will be eliminated and data deviated from the surrounding data patterns will be assigned lower density values. Next, we will use examples to show both theoretically and experimentally that reference-based method can successfully identify local outliers as well as global outliers in complex datasets that contain clusters of different densities.

we generated a small dataset using the synthetic data generating system that we implemented to automate the generation of various datasets. Details of the synthetic data generator will be discussed in the next chapter. The generated dataset X contains 850 2D data points.

As shown in Figure 3.3, there are three clusters $C1, C2$ and $C3$, where data in clusters $C1$ and $C2$ are uniformly distributed and data in $C3$ are in normal distribution. Cluster $C1$ has a lower density compared to $C2$ and $C3$. In addition to the three clusters which form the main body of the dataset, there are a few local outliers as well as some global outliers in the given dataset. In our experiment, we set the number of reference-based nearest neighbors $k = 4$. To ensure that the reference points are evenly positioned in the 2D data space, we set the number of reference point to be 16 which is 4 to the power of the dimensionality so that each axes is



(a)

(b)

Figure 3.4: Finding top eight outliers (a) using reference-based method and (b) using the traditional KNN method

divided evenly. In the first run to mine the top 3 outliers, the three global outliers o_1, o_2 and o_3 are found. Since we want to check if the local outliers specific to the clusters can be found, the program is set to mine the top eight outliers in the second run. The result is displayed in Figure 3.4 (a), where the outliers are marked with a cross.

We tested the distance-based outlier detection method with the same dataset. The implementation is based on Definition 3.2.2, where the neighborhood density $D^m(x, k)$ is the reciprocal of the distance to the k^{th} nearest neighbor. As above, we set $k = 4$. The top 3 outliers identified are the same as those identified using our method. However, it fails to find the two local outliers specific to cluster $C2$ when mining the top 8 outliers. The result is shown in Figure 3.4 (b), where the undetected local outliers are marked with arrows. It is observed that $D^m(x, k)$ method tends to consider data in sparse regions as outliers such as those located in the perimeter of cluster $C1$. The ranking result shows that using $D^m(x, k)$ method, o_5 is ranked in the 9^{th} place but o_4 is ranked in the 76^{th} place. In order to find the local outlier o_4, the distance-based method has to falsely mark many other data points as outliers.

This actually poses an interesting question: what will happen if the number of real outliers in a dataset is less than the number of top outliers the program is set to mine? Such issue can be important with large real world datasets whose domain knowledge is unknown. We test both reference-based approach and the distance-based approach to mine the top 76 outliers, when o_4 can only be identified using the distance-based method. The result is shown in Figure 3.5, where (a) is the result using the reference-based method and (b) is the one using the distance-based method. We can see that in Figure 3.5 (b) many data points deep in the relatively sparse cluster $C1$ are falsely marked as outliers before distance-based method is able to find the real local outlier o_4. Although there are a few points inside cluster $C1$ are marked as outliers by our approach, their rank are lower than those real outliers. Careful observation shows that they are the next best outliers within $C1$ compared to those points on the border of $C1$. The overall outliers identified indicate that the reference-based approach is more likely to consider the data points deviating from or lying on the edge of the data patterns as outliers.

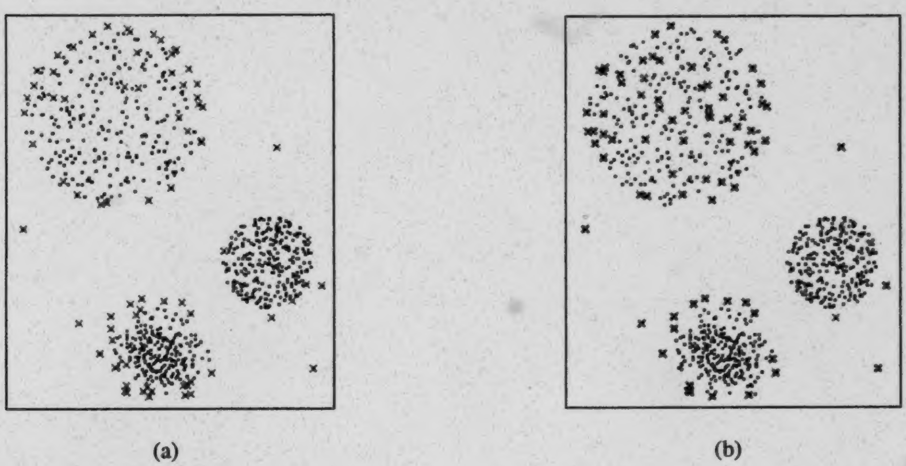

(a) (b)

Figure 3.5: Finding top 76 outliers (a) using reference-based method and (b) using the traditional KNN method

The above examples demonstrate that compared to distance-based approach, reference-based method is not only superior in differentiating data deep inside a sparsely distributed cluster from local outliers deviated from a dense pattern in a dataset, but also capable of eliminating false identification of outliers inside the

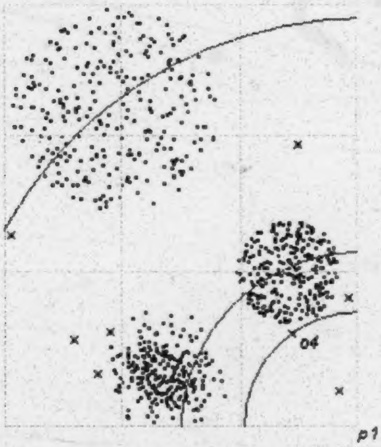


Figure 3.6: Finding local outliers using reference-based approach

sparse data patterns of a complex dataset.

One of the intentions of the reference-based approach is to best approximate the distance measure obtained using the traditional k nearest neighbor approach so that global outliers can be effectively identified but in a rather efficient way. In this sense, the more the number of reference points, the better the distance approximation and the more accurate the identification of global outliers in a dataset. However, when the whole dataset is used as the set of reference points, reference-based approach is reduced to the traditional distance-based approach, which can lead to false identification in detecting local outliers in a complex dataset. While this is an intrinsic problem with the existing distance-based approach, it can be easily solved in our reference-based method by starting at a few number of reference points and then incrementally increasing the number of reference points. Since the reference-based nearest neighbors of the data points computed with respect to a given set of reference points can be reused for the next round when more reference points are added, such adjustment will have little impact on the performance of the outlier detection method. The detection process stops when certain outliers have been found and such inspection often involves human intervention.

To further explain why a small number of reference points can facilitate the identification of local outliers, we use the previous dataset as an example. Suppose we have a set of 16 reference points $P = \{p_1, ..., p_{16}\}$, which are the vertices on a grid obtained by partitioning the axes in the data space as shown in Figure 3.6. Take the reference point p_1 as an example. If we draw circles with p_1 as the center and the distance to each data point as the radius, the k reference-based nearest neighbors of a data point x with regard to p_1 would be those on or closest to the circle where x is located. As is shown in the plot, there are no other data points that fall on the same circle as the local outlier o_4 does. In contrast, data deep in clusters $C1, C2$, and $C3$ generally have nearest neighbors with little or no distance difference. Consequently, o_4 will have a relatively smaller neighborhood density in the one dimensional data space X^{p_1} than those data in the clusters. Though with another reference point, say, p_2, it is possible that o_4 may be assigned a high density if the circle (p_2 as the center and $d(p_2, o_4)$ as the radius) pass through one or more dense data patterns, the overall neighborhood density is determined by the minimum among all the calculated densities for x in terms of X^{p_r}. Assume that the reference points are sparsely distributed such that each data point in the data patterns are not isolated by the circle around a reference point, then data deep in clusters are ensured to have very close neighbors which contribute to high neighborhood densities with regard to each reference point. Therefore, o_4 will have a lower neighborhood density and a higher outlier score than data in the clusters.

One should also notice that there is indeed a tradeoff between the number of reference points and the ability of the reference-based method to detect global and/or local outliers. On the one hand, if all the data points in the original dataset are used as the reference points, then our approach reduces to the traditional KNN approach. On the other hand, by using a small set of reference points, local outliers will have a better chance of being detected at the potential cost of some inaccuracy in the overall quality of all the detected outliers. We leave it as a future research topic to investigate how to achieve such a tradeoff and how to integrate our approach with the various approaches to local and/or global outlier detection in the literature such as LOF [13].

3.4 Empirical Evaluation

In this section, we show experimentally that the proposed method can efficiently identify local and global outliers in various datasets. We compare the performance of our approach with the existing KNN-based approaches, including distance and density-based methods.

3.4.1 Results on Synthetic Datasets

To compare the performance of the proposed reference-based approach with the existing KNN-based approach, our first test is to see how fast each method can find outliers in large datasets. Since it is well accepted that the evaluation of outlier detection involves human intervention, we limit our experiments to two-dimensional synthetic data so that the evaluation can be performed by simple visual inspection. Using our synthetic data generating system we generated a set of synthetic datasets. The size of these datasets ranges from 1,000 to 500,000 data points. To be consistent, each dataset has a major data pattern that is normally distributed. Based on the fact that outliers accounts for only a very small portion of data in a dataset, the number of outliers to be mined is set to be 1% of the data size in all the following experiments. For the reference-based approach, the number of reference points can be set to a constant for all these datasets. This is because all the test datasets have similar probability distribution and each of them contains only one normally distributed cluster. In such cases, a few reference points that cover different areas in the data space are sufficient to ensure the successful detection of outliers. As discussed before, the reference points are evenly located in the grid vertices of the data space. In our experiment, the number of reference points is set to 9 for all these datasets. We implemented the reference-based method in Java to facilitate the visualization of the outlier detection results. For the KNN-based method, we downloaded the executable version of Orca, the C implementation of the distance-based method discussed in [8] from the author's website. Since Orca is based on the distance-based algorithm that has near linear performance, it is believed to be one of the fastest KNN-based outlier detection method. To compare the two programs,

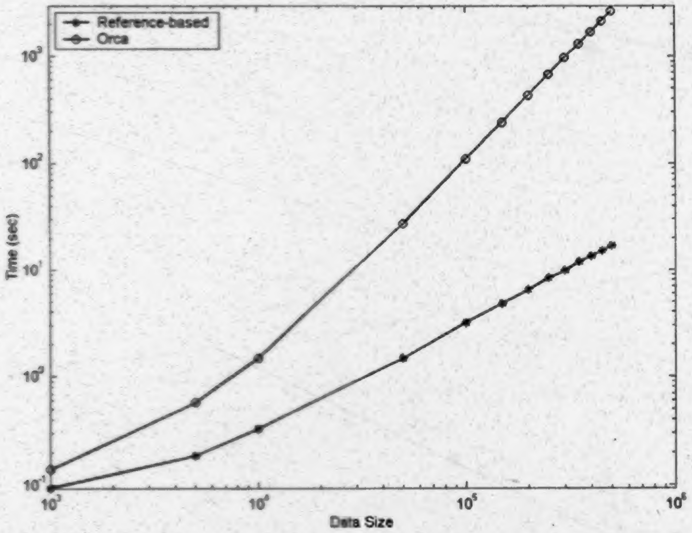


Figure 3.7: Log-scale execution time vs data size for reference-based approach and Orca

we run our method in command line mode so that the execution time includes data loading and writing results to the standard output as does Orca. Orca also requires preprocessing the data to randomize the order of the original data and then convert the data to binary format prior to outlier detection. In our experiments, the cost of data preprocessing for Orca is not counted in the recording of Orca's execution time. The number of nearest neighbors is set to be $k = 6$ for both programs.

Since each dataset contains a single cluster that is normally distributed, both programs can effectively mark the data lying farther away from the mean as outliers in a dataset. There is hardly any difference in the identified outliers using the two methods. However, the difference of execution time between our reference-based approach and Orca gets bigger and bigger with the increase of the data size despite the fact that our implementation of ROS is in Java while Orca is in C. It is easy to see that the execution time for ROS is a function of n only since R has a constant value for all the datasets. Figure 3.7 is the log-scale plot of execution time vs data size for the two methods. Although the plot for Orca does not include the data preprocessing time, Figure 3.7 shows that with large datasets, reference-based approach has orders of magnitude improvements in execution speed compared to Orca, the optimized implementation of the distance-based approach. It is also worth noting that the execution time of the reference-based approach is near linear with the increase of the data size while the execution time of Orca tends to be near linear only from a certain point, i.e., data size has to be sufficiently big. The results further demonstrate that our approach is very efficient and highly scalable to very large datasets.

Outliers are defined as data points that deviate from the main patterns of a dataset. They are most likely to be considered in the context of clusters with different types of data distributions. That is, an object is marked as an outlier if it is isolated from the clusters in a given dataset. To test if the reference-based approach can effectively find meaningful outliers in a complex dataset, we generated a dataset of 10,000 datapoints. There are six data patterns. Three of them are uniformly distributed and the other three are normally distributed, but they all have different densities.

Unlike the datasets containing only normally distributed data where outliers lie on the outer fringe of each cluster, this dataset has two types of outliers: outliers uniformly distributed around the main data patterns and outliers lying on the outer skirt of the three normally distributed clusters. To identify outliers in such complex datasets, the value for R, the number of reference points is usually higher than that for simple datasets since relatively more view points inside the data space can achieve better *view* of the data. As being discussed, the number of reference points for can be decided by incrementally fine tune the resolution and we set the value to be 196. To demonstrate the effectiveness of reference-based approach in finding both global and local outliers, we implemented LOF according to [13] that has been well known for its performance in detecting local outliers. We run the three methods: distance-based, LOF and our ROS. For distance-based and ROS, we set $k = 4$. For LOF, we set $MinPts = 30$ as recommended in [13].

Figure 3.8, 3.9 and 3.10 are the screen shots showing the results for mining the top 100 (1% of data size) outliers. In all three graphs, outliers are marked with crosses. Visual inspection shows that the distance-based method, LOF, and ROS

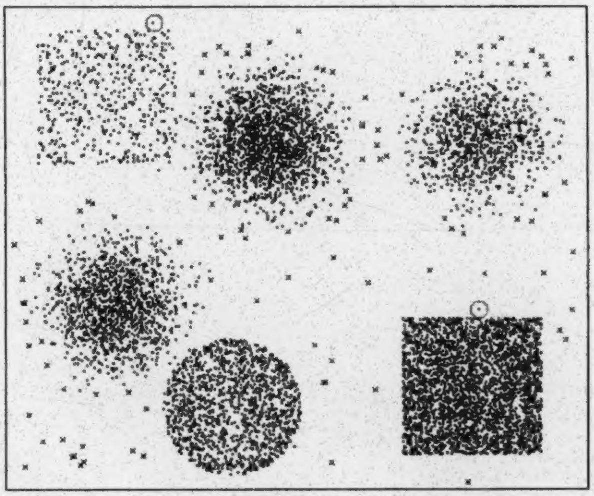


Figure 3.8: Outlier detection result from KNN-based approach

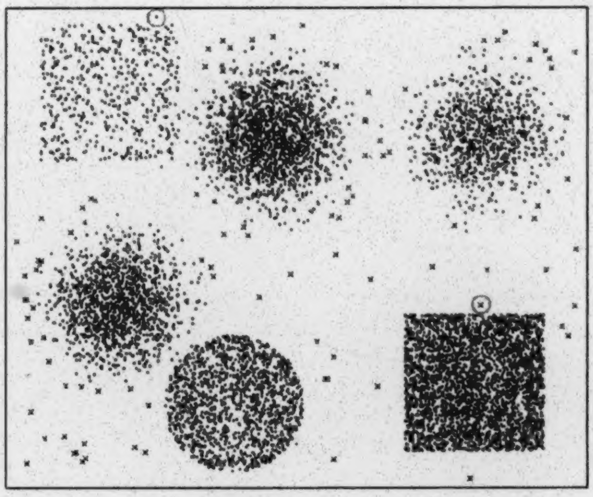


Figure 3.9: Outlier detection result from LOF

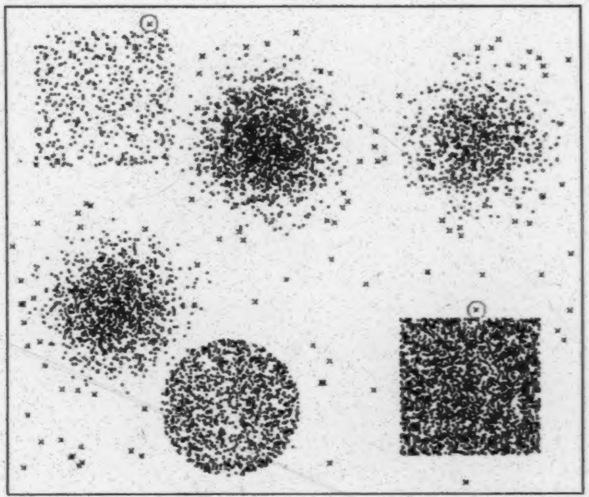


Figure 3.10: Outlier detection result from ROS

are all able to discover the global outliers. Furthermore, two local outliers (the two crosses surrounded by small light-colored circles in Figure 3.10) are discovered by our reference-based method ROS, while LOF can only find one of them and the distance-based method fails to find either of them. Therefore, our approach is not only effective in finding outliers that lie far away from the major patterns in the dataset, it also achieves similar or even better results than LOF in detecting local outliers.

3.4.2 Results on Hockey Data

The National Hockey League (NHL) data have been used as benchmark testing datasets in several outlier analysis works [45, 13, 60]. We use the statistics of NHL 2003-2004 season obtained from the NHL website[54]. The dataset contains 916 entries. For performance evaluation, we compare our approach with both Orca and LOF. The goal is to see if our approach can efficiently find the outliers identified by Orca and/or LOF in real-world datasets. Given a 3D dataset, which is moderate in dimensionality, we can build a cube in the 3D space with all data inside the cube and set 8 reference points that are located on the vertices of the cube. We set $k = 4$ for both Orca and our approach and $MinPts = 30$ for LOF to ensure LOF finding

Table 3.1: Outlier detection result 1 on NHL(03/04) data

Our Rank	**LOF Rank**	**Orca Rank**	**Player**	**Games Played**	**Goals scored**	**Shooting Percentage**
1	1	1	Milan Michalek	2	1	100
2	2	2	Pat Kavanagh	3	1	100
3	3	3	Lubomir Sekeras	4	1	50
minimum				*1*	*0*	*0*
median				*57*	*4*	*6.6*
maximum				*83*	*41*	*100*

Table 3.2: Shooting percentage on NHL(03/04) data

Shooting Percentage (goals/shots)	0-10	10.1-20	20.1-30	30.1-40	41.1-100
Number of Players	654	235	18	6	3

meaningful outliers.

We conduct the experiment in a similar way as other outlier analysis works. It consists of two tests. The first test mines the outliers based on the three attributes: games played, goals scored and shooting percentage. LOF, Orca and our reference-based approach achieve identical results and the top three outliers are listed in Table 3.1.

The outlier status of the three identified players are obvious. They only played a few games and scored once. But their shooting percentage are unusually high as explained by Table 3.2.

The second test is to mine outliers based on the three attributes: points scored, plus-minus statistic and penalty minutes. The top 3 outliers found by our reference-based approach are listed in Table 3.3. Sean Avery is on top because his points and plus-minus figures are moderate but the number of penalty minutes is the highest among all the players. LOF gets similar result to ROS while Orca's result is slightly different. The top rank from Orca, Zdeno Chara, is ranked as the 9th outlier by our approach and 26th outlier by LOF. Careful investigation shows that outliers identified by Orca tend to be in a sparse region regardless of the data distribution. Our reference-based method instead favors outliers that deviate from the main data patterns and the results are close to LOF in this aspect. For example, Jody, who is ranked third by our approach lies far away from the trend of the data body due to his low points and plus-minus statistic but extremely high penalty minutes. With 3D datasets of size about 1,000, the execution time for each of these methods is within 0.1 seconds and can be neglected.

Table 3.3: Outlier detection result 2 on NHL(03/04) data

Our Rank	**LOF Rank**	**Orca Rank**	**Player**	**Points Scored**	**Plus-Minus**	**Penalty minutes**
1	1	3	Sean Avery	28	2	261
2	2	2	Chris Simon	28	15	250
3	7	15	Jody Shelley	6	-10	228
⋮	⋮	⋮	⋮		⋱	
9	26	1	Zdeno Chara	41	33	147
minimum				*0*	*-46*	*0*
median				*12*	*-1*	*26*
maximum				*94*	*35*	*261*

3.5 Conclusion

In this chapter, we have proposed an efficient reference-based outlier detection method that uses the relative degree of density with respect to a set of reference points to calculate the neighborhood density of a data point. In addition to being compatible with the traditional distance-based outlier detection methods, our approach performs better in identifying local outliers that deviate from the main patterns in a given dataset. The execution time of our algorithm is $O(Rn \log n)$ where n is the size of dataset and R is the number of reference points. Candidate outliers are ranked according to ROS that has been assigned to each data point. Theoretical analysis and empirical studies show that in addition to being highly efficient and scalable to very large datasets, our method can detect both global and local outliers. When all the data points are used as references points, our method becomes a distance-based approach finding global outliers. When less reference points are used, local outliers relative to the data patterns are discovered. We advocate the use of a small number of reference points uniformly distributed over the data (using a grid) to find global and local oultiers effectively and efficiently.

Chapter 4

A Synthetic Data Generator for Clustering and Outlier Analysis

We present a distribution-based and transformation-based approach to synthetic data generation and demonstrate that the approach is very efficient in generating different types of multi-dimensional numerical datasets for data clustering and outlier analysis. We developed a data generating system that is able to systematically create testing datasets based on user's requirements such as the number of points, the number of clusters, the size, shapes and locations of clusters, and the density level of clusters and noise/outliers in a dataset. Two standard probability distributions are considered in data generation. One is uniform distribution and the other is normal distribution. Since outlier detection, especially local outlier detection, is conducted in the context of clusters of a dataset, our synthetic data generator is suitable for both clustering and outlier analysis. In addition, the data format has been carefully designed so that the generated data can be visualized not only by our system but also by some popular statistical rendering tools such as statCrunch [68] and statPoint [69] that display data with standard statistical graphical approaches. To our knowledge, our system is probably the first synthetic data generation system that systematically generates datasets for evaluating the clustering and outlier analysis algorithms. Being an object-oriented system, the current data generator can be easily integrated into other data analysis systems.

4.1 Introduction

Clustering analysis and outlier detection are two important techniques widely used in data mining and automatic knowledge discovery. Although research on outlier analysis is relatively a new topic in the area of data mining as compared to data clustering, they have both been addressed by many researchers and there exist a large number of approaches to clustering and outlier analysis. While different algorithms have their own strength in finding clusters and/or outliers, the performance of a particular algorithm can be quite different on different datasets. Therefore, the choice of clustering or outlier analysis methods depends on the specific purpose of the application as well as the datasets available. This in turn poses one of the most important issues in data analysis: How do we assess a data analysis algorithm?

It is hard to say that one algorithm is better than the other since different algorithms usually use different testing datasets with certain constraints such as data distribution, dimension and density in the analysis of the effectiveness and efficiency. There exist some databases with a variety of datasets obtained from real life environment. These datasets could be in various formats and distributions that make it difficult to use them in testing and comparing different clustering and/or outlier algorithms. Surprisingly, little work has been done on systematically generating artificial datasets for the analysis and evaluation of data analysis algorithms in data mining area.

In this work, we explore the idea to automatically generate datasets in two or more dimensional space given the total number of points N and the number of clusters K in a dataset. We use data points to represent objects with multiple attributes. The properties of each dataset, including the space between clusters, the cluster distributions and outlier densities are specified by the user but controlled automatically by the system. Each dataset is generated along with a difficulty level, a density level, an outlier level and a certain data distribution. Given a fixed number of points in a dataset, the size and density of clusters are closely related and are both controlled by the density level. The spreading and density of outliers with respect to the main body of the data are determined by the outlier level. The difficulty level is defined in terms of the existing clustering algorithms and they are roughly classified into three groups:

- easy level - the datasets at this level have only spherical or convex clusters;
- medium level- the datasets have long thin or arbitrarily shaped clusters;
- difficult level - the datasets can have clusters within clusters with all possible shapes.

The data generator can be used not only in the evaluation and testing of data clustering analysis and outlier detection but also in visualizing various data distributions . Our goal is to develop a general framework for the generation of testing datasets with controlled level of clustering difficulties and devise a heuristic that can be improved upon in a meaningful way in high-dimensional and categorical space in the future. We investigate current research and implementation on data generation and proceed in different stages. An important part of data generation is to display the produced datasets in a graphical user interface for visual inspection. Hence, we combine the algorithm design with the implementation together in each stage of the development. Several methods such as distribution-based approaches and transformation-based approaches, or their combination have been employed in generating meaningful datasets. Java Swing is used as the programming language since the implementation of the data generation system relies heavily on the graphical user interface (GUI). In addition to generating datasets that satisfy the user's specific requirements and displaying the data in a GUI for visual inspection, the system has other functionalities including saving and exporting the generated datasets to local files as well as importing and visualizing the existing data.

4.2 Existing Work on Synthetic Data Generation

An important issue in evaluating data analysis algorithms is the availability of representative data. When real-life data are hard to obtain or when their properties are hard to modify for testing and comparing various algorithms, synthetic data become an appealing alternative. Most existing work on clustering and outlier analysis uses both synthetic data and real-life data to test the validity and performance of the proposed algorithms.

Data generation has been an important topic in mathematics and statistics. There are some state-of-the-art techniques on generating data of certain distribution, for example, random sequences and normal distribution, which serve as the fundamental tools for synthetic data generation systems in many applications. Despite increasing interest, the research on synthetic data generation in the area of data mining is still in its early stage. There exist some well-known datasets that have been widely used as benchmark datasets to test the performance of many clustering algorithms. Among them, one is provided by the team that developed the clustering algorithm CHAMELEON [42]. The dataset has 10,000 2D points and includes not only different shapes of clusters but also different type of outliers. Unfortunately, there is no description of how these datasets are generated.

In the literature of software testing, a large number of methods to automate test data generation have been studied [18]. In recent years, research areas such as data mining [38], sensor networks [77], artificial intelligence [62] and bioinformatics [74] are paying more attention to the development of data generation systems to systematically generate synthetic data for numerous applications. In this chapter, we will briefly discuss some existing data generation methods and systems.

4.2.1 IBM Quest Synthetic Data Generator

A well-known synthetic data generation system is developed by the IBM's QUEST data mining group [38]. The system consists of two data generators. One is used to generate transaction data for mining associations and sequential patterns. Given some parameters, the system can produce a set of data containing information of customer transactions. The other generator produces data intended for the task of classification. The output is a person database in which each entry has nine attributes. QUEST also developed a series of classification functions of increasing complexity that use the nine attributes to classify people into different groups.

The generated datasets contain only numerical values. Values of non-numerical attributes are converted to numerical values according to some pre-defined rules.

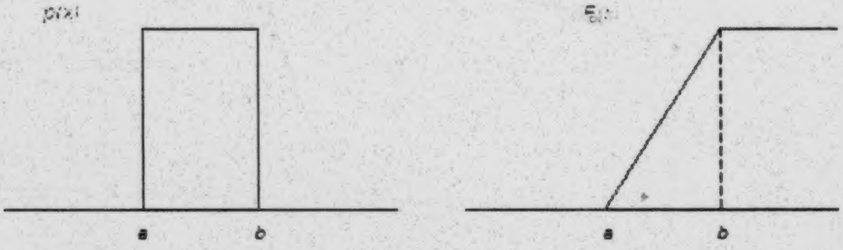
4.2.2 Synthetic Data Generation in Other Research Fields

Synthetic data generation also plays an important rule in many different fields of computer science such as Information retrieval, software engineering and artificial intelligence, although in each field the focus and the requirement of generating synthetic data are quite different.

The GSTD algorithm proposed in [72] uses three operations to generate spatiotemporal datasets by gradually altering the three parameters that control the duration, the location, and the size of spatiotemporal objects. Such a data generator serves as an integral part of the benchmark environment for spatiotemporal data access system.

The main focus of test data generation in automatic software testing is to generate input data to test the correctness of a given computer program or software system. To have a sufficient coverage on the execution of a computer program, a data generation system first needs to analyze the control flow of the program to identify target execution paths to be tested. Input data with which the execution of the program follows a specific path are usually generated by using either symbolic evaluation techniques or solving a properly formulated optimization problem.

In the field of artificial intelligence, many important problems are NP-hard such as the Boolean satisfiability problem (SAT). To test the performance of solvers and algorithms for these problems, one also needs to generate testing problem instances. In addition to real-world and manually compiled benchmarks, a recent trend is to generate problem instances randomly from some probability distribution. As a matter of fact, the study of the typical-case hardness of randomly-generated problem instances and the performance of various algorithms on these instances has been an important research topic in artificial intelligence. On the one hand, many deep theoretical results on the complexity of NP-hard problems and useful insights into the design of more efficient algorithm have been obtained. On the other hand, hard testing problem instances generated at the so-called phase transition region of some random problem model have been one of the driving forces in the development of the start-of-the-art solvers for these AI problems.



(a) (b)

Figure 4.1: PDF and CDF of uniform distribution

4.3 Mathematical Tools and Techniques

In an effort to systematically generate test datasets for data analysis, we make use of some mathematical tools such as probability distributions and linear transformations. By applying these tools, the proposed method provides the mechanism that datasets are not only generated automatically but also controlled by the parameters from the user input. This section introduces the mathematical concepts and tools related to our proposed approach.

4.3.1 Uniform Distribution

The uniform distribution is the simplest continuous distribution in probability. A random variable x has the uniform distribution if all possible values of the variable are equally probable [61]. It is also called rectangular distribution.

Uniform distribution is specified by two parameters: the end points a and b. The distribution has constant probability density on the interval (a, b) and zero probability density elsewhere. The probability density function(PDF) and cumulative distribution function(CDF) for a continuous uniform distribution on (a, b) are

$$f(x) = \begin{cases} \frac{1}{b-a}, & a < x < b; \\ 0, & \text{otherwise.} \end{cases}$$

$$F(x) = \begin{cases} 0, & x < a; \\ \frac{x-a}{b-a}, & a < x < b; \\ 1, & x > b. \end{cases}$$

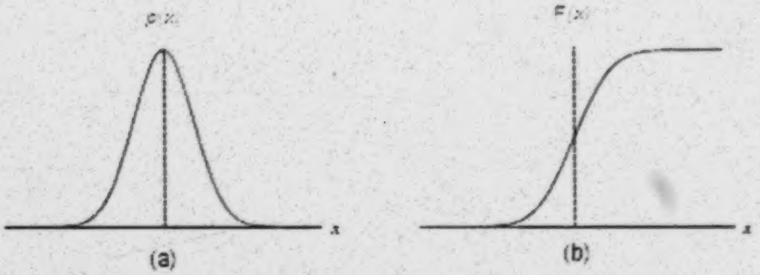


Figure 4.2: PDF and CDF of normal distribution

In Figure 4.1, (a) is a plot of the uniform PDF and (b) is a plot of the uniform CDF. The standard uniform distribution is the case where $a = 0$ and $b = 1$.

We aim to generate data from a multivariate uniform distribution. The dataset D is composed of a set of multi-dimensional points. Each point in $D = \{x_1, x_2, ..., x_m\}$ is obtained by generating uniform random numbers for x_i, where $i = 1, 2, ..., m$. The attribute values of each variable are uniformly distributed in $(0, 1)$. Since the joint distribution of two or more independent one-dimensional uniform distributions is also uniform, the points in D are uniformly distributed in the feature space of all variables.

4.3.2 Normal Distribution

A continuous random variable x has a normal distribution or Gaussian distribution if its probability density function is

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-(x-\mu)^2/2\sigma^2},$$

where μ is mean, σ^2 is the variance and $-\infty < x < \infty$ [61].

Figure 4.2(a) is the plot of the normal PDF and Figure 4.2(b) is the plot of the normal CDF respectively. Standard normal distribution is the normal distribution given $\mu = 0$ and $\sigma^2 = 1$.

In our implementation, covariance matrix that controls the attributes of data points has been widely used to generated various shaped normal distributions.

4.3.3 Box-muller Transformation

Box-Muller transformation allows us to transform a two-dimensional continuous uniform distribution to a two-dimensional bivariate normal distribution (or complex normal distribution) [52]. Let x_1 and x_2 be two independent random variables uniformly distributed between 0 and 1. The basic form of Box-Muller transformation is defined as

$$y_1 = \sqrt{-2\ln x_1} cos(2\pi x_2),$$
$$y_2 = \sqrt{-2\ln x_1} sin(2\pi x_2),$$

where y_1 and y_2 have a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.

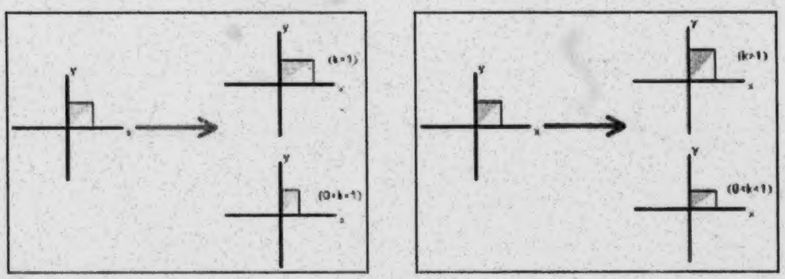
In our data generation system, rather than using the normal cumulative distribution function to generate normal distributions, which does not have an explicit expression, we adopted Box-muller transformation. By applying the above formulas, we are able to transform uniformly distributed random variables x_1 and x_2 to two random variables y_1 and y_2 with a joint normal distribution.

4.3.4 Linear Transformation

A linear transformation between two vector spaces U and V is a mapping $T : U \longrightarrow V$ such that

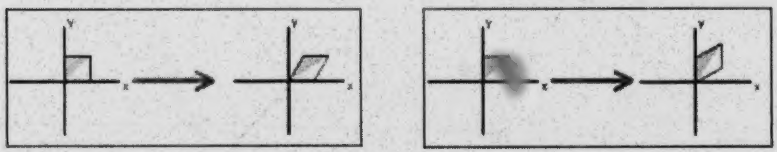
1. $T(u_1 + u_2) = T(u_1) + T(u_2)$, for any vectors u_1 and u_2 in U,
2. $T(\alpha u) = \alpha T(u)$, for any scalar α and arbitrary vector u in U.

Suppose $U = R^2$ and $V = R^2$, $T : R^2 \longrightarrow R^2$ is a linear transformation if and only if there exists a 2×2 matrix A such that $T(u) = Au$ for all u in R^2 [36]. Matrix A is called the standard matrix for T. Linear transformation in two dimensional vector space has been extensively used in our data generation system to dynamically produce two dimensional datasets of various characteristics. Once we have obtained the *basic dataset*, which will be detailed in section 4.4, we can control the shape, density and location of each cluster in the output dataset by applying to each vector/point in the basic dataset linear transformations such as shears, reflections,



(a) (b)

Figure 4.3: Linear transformation: expansions and contractions



(a) (b)

Figure 4.4: Linear transformation: shears

contractions, expansions and translations. The linear transformation of a normal distribution is still a normal distribution, but the linear transformation of a uniform distribution is not necessarily a uniform distribution.

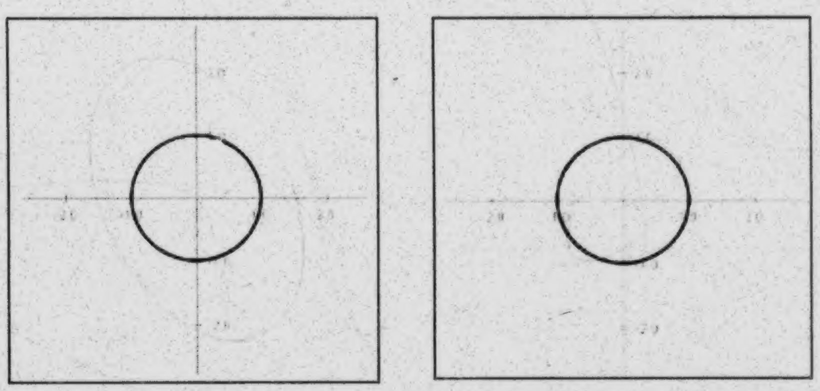
Figure 4.3 and 4.4 are examples used in [36] to illustrate the action of a linear transformation $T : R^2 \to R^2$. The image of a unit square under T is employed to demonstrate the geometric meaning of different types of linear transformation.

Figure 4.3(a) indicates how expansion and contraction along x-axis work. Given a set of column vector $[p_x \;\; p_y]^T$, expansion and contraction along x-axis is given by the standard matrix

$$A = \begin{bmatrix} k & 0 \\ 0 & 1 \end{bmatrix}.$$

Thus, the vectors "stretch" along the x-axis to $[kp_x \;\; p_y]^T$ for $k > 1$ and "compress" along the x-axis for $0 < k < 1$.

Similarly, Figure 4.3(b) is an example showing the expansion and contraction of the unit square along y-axis. The standard matrix used here is



(a) (b)

Figure 4.5: Linear transformation examples

$$A = \begin{bmatrix} 1 & 0 \\ 0 & k \end{bmatrix}$$

which takes the vectors $[p_x \ \ p_y]^T$ to $[p_x \ \ kp_y]^T$. In this case, the standard matrix A stretches the vector along y-axis when $k > 1$ and compresses it along y-axis when $0 < k < 1$.

A shear in the x-direction is shown in Figure 4.4 (a). It is achieved using the standard matrix

$$A = \begin{bmatrix} 1 & k \\ 0 & 1 \end{bmatrix}$$

to convert vectors $[p_x \ \ p_y]^T$ to $[(p_x + kp_y) \ \ p_y]^T$.

A shear in the y-direction is given in Figure 4.4 (b), in which the standard matrix

$$A = \begin{bmatrix} 1 & 0 \\ k & 1 \end{bmatrix}$$

is used taking $[p_x \ \ p_y]^T$ to $[(p_x + kp_y) \ \ p_y]^T$.

To generate datasets with various patterns and densities, we often use a more complicated standard matrix to transform a set of data. The operation can be considered as the composition of several linear transformation, such as a rotation, a magnification, and a translation. A typical example is shown in Figure 4.5 where a unit circle is transformed into an enlarged oval as in (a) and a contracted oval as in (b), where the standard matrices leading to these transformations are

$$A = \begin{bmatrix} -1.2 & 1.1 \\ 2 & 1 \end{bmatrix},$$

and

$$A = \begin{bmatrix} 0.3 & 0.8 \\ 0.9 & 0.3 \end{bmatrix}$$

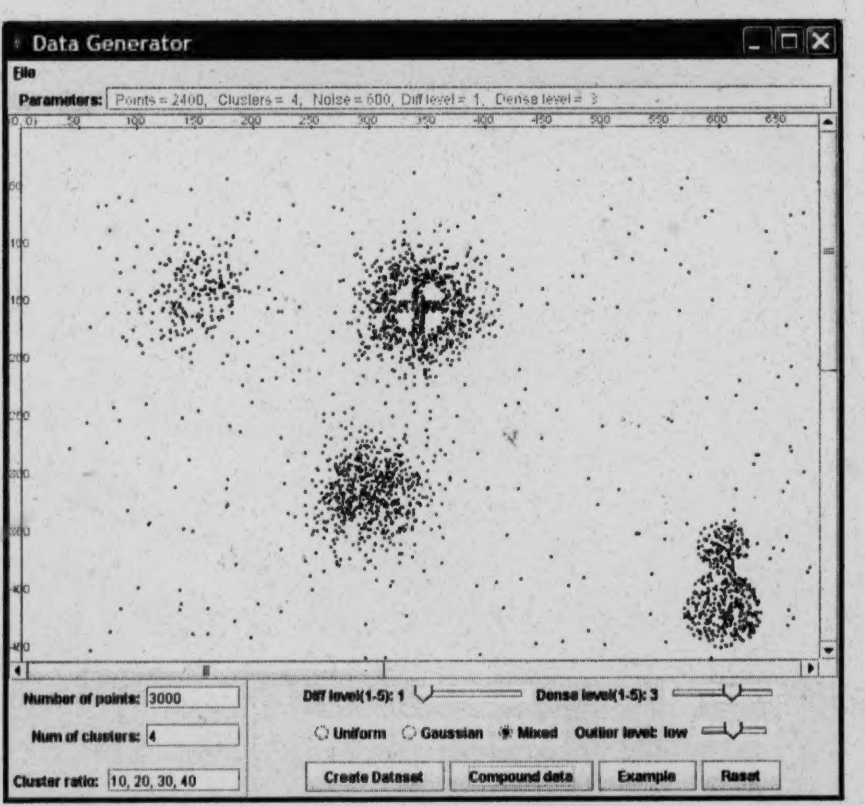
respectively. The transformed ovals may be shifted to a different location by translations through vector addition.

4.4 A Comprehensive Approach to Synthetic Data Generation

In this section, we present a hybrid approach to synthetic data generation. The proposed approach is aimed at providing a basic modelling framework for generating data that can be used to evaluate and test clustering and outlier analysis algorithms.

It has been well recognized that the performance of different data analysis algorithms depends heavily on the testing datasets. Among the existing clustering algorithms, the partitioning methods can easily identify clusters with spherical shapes, but they are unable to find clusters of irregular shapes and tend to split an elongated cluster into different groups. Although the density-based methods can handle clusters of arbitrary shapes and various sizes, they are very sensitive to the density of each cluster in a given dataset, which may lead to failure in detecting clusters with unevenly distributed data. Since outliers are data that deviate from the main pattern of a dataset, they are always considered in the context of clusters. That is, an object is marked as an outlier if it is isolated from the clusters in the dataset. The causes for such isolation can be generalized in two categories: (1) outliers are located in a less dense region compared to the density of the clusters; and (2) outliers do not fit into the cluster patterns. Therefore, outlier detection, especially local outlier detection that defines outliers with respect to the neighborhood density and patterns is often conducted by differentiating them from data in clusters.

In our method of synthetic data generation, each output dataset is specified by a difficulty level, which is defined in terms of data distributions and cluster shapes.



Data Generator
File
0, 0
50
100
150
200
250
300
350
400
450
500
550
600
650
Create Dataset
Compound data
Example
Reset

Parameters: Points = 2400, Clusters = 4, Noise = 600, Diff level = 1, Dense level = 3

Number of points: 3000

Num of clusters: 4

Cluster ratio: 10, 20, 30, 40

Diff level(1-5): 1

Dense level(1-5): 3

○ Uniform ○ Gaussian ● Mixed

Outlier level: low

Figure 4.6: A screen shot of the synthetic data generation system

Since the difficulty level of a dataset indicates the complexity in identifying clusters, it provides us a measure of how a clustering algorithm works. Apart from the difficulty level, each dataset is also assigned a density level and a noise level. Like the difficulty level, the noise level is used to define the distribution and density of outliers or noise. Other parameters from user input are the number of points, the number of clusters and the percentage of points for each cluster in a dataset. The created data objects are represented by points, e.g., points in two dimensional space with x and y being floating point numbers. We built a graphical user interface to display the generated 2D dataset for visual inspection. Figure 4.6 is a screen shot of the synthetic data generation system. As is shown in the figure, the shape and density of the output clusters as well as the distance between the means of different clusters in a dataset are determined by the standard distribution, the difficulty level and the density level.

To automate the data generation process, the system proceeds in two steps. The first step is to create the *basic dataset*, in which the data in each cluster have a standard distribution. For the uniform distribution, the x and y values of all the points in the basic dataset are in $(0, 1)$. For normal distribution, the basic dataset contains clusters that have a standard normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. The second step is to apply some mathematical techniques to generate the required dataset. Once we have the basic dataset, three major methods are used in creating clusters and outliers with different shapes and densities.

- Linear transformation, which involves matrix multiplication to translate, shear, contract or expend the the clusters in the basic dataset.
- Linear equation, which controls the line-shaped clusters and outliers.
- Circle equation, which controls the curve-shaped clusters.

The technical details of synthetic data generation will be presented in two aspects. One is the dynamic control and generation of clusters. The other aspect is about how the outliers are distributed. To make the concept concrete to the readers, a visual approach is taken in presenting the method.

4.4.1 Generation of Clusters in a Dataset

The generation of clusters of a dataset involves the determination of cluster densities, sizes, shapes and relative locations. By analyzing the user input, the synthetic data generation system automatically controls all these aspects. Two parameters *density level* and *cluster ratio* are the major factors to contribute to the density and size of each cluster in a dataset. Given a density level, an appropriate standard matrix is calculated to transform the basic clusters [1] into ones with either expanded or contracted sizes. The higher the density level, the smaller the cluster size and the more compacted the data in the clusters. By default, data are evenly distributed to each cluster in a dataset. For example, if dataset D has 1,000 data objects that form 4 clusters, the system would automatically assign 250 data to each cluster. The parameter cluster ratio provides the user with an option to set the number of data objects for each cluster. It consists of a sequence of integers indicating the percentage of data in each cluster over the total number of data in a dataset. By parsing the cluster ratios, the system adjusts the number of data in each cluster to satisfy the user's specific requirements. This, in turn, will change the density of each cluster since each cluster size remains unchanged.

Cluster shapes and relative locations are mostly determined by the parameter difficulty level. In the following, we will discuss the generation of datasets classified into five difficulty levels based on the distribution of the data in clusters. Given a difficulty level, the specific locations and shapes of the clusters in a dataset is controlled by the system in a random manner, i.e., the cluster can have any of the shapes belonging to this difficultly level and lie in any region in the dataset. The distances between clusters are checked to ensure that clusters are not overlapping. This is especially important for simple datasets with low difficulty levels. Alternatively, a dataset may consist of randomly produced clusters from different difficutly levels when one prefers to have a sophisticated set of data. Therefore even with identical parameter sets, there are hardly any datasets that are exactly the same due to the randomness in deciding cluster locations and shapes. Apart from being visualized, the generated data can be saved to a file in case that the same data are required for

---

[1] Attribute values in such clusters are usually in $(0, 1)$

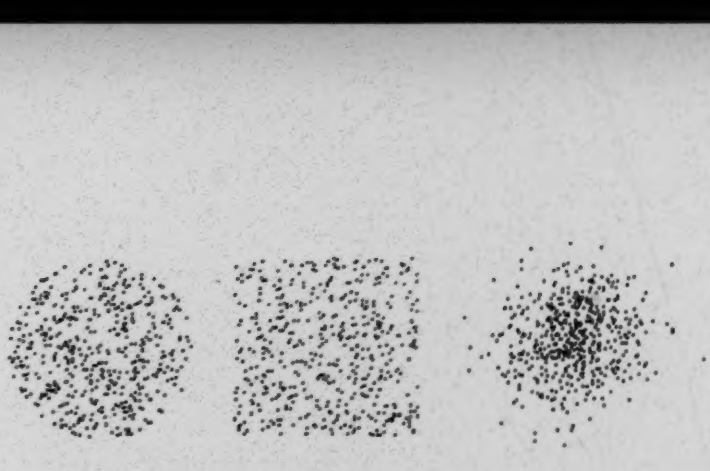


Figure 4.7: Difficulty level 1: each cluster contains 500 2D points

later inspection or testing different data analysis algorithms.

Datasets with Difficulty Level One

The datasets at this level are the simplest in terms of the definition of clusters. There are two major features of the clusters in such a dataset.

- All clusters have only spherical or square shapes.
- Clusters are well separated.

Following the generation of the basic datasets, the transformation of contraction and/or expansion are applied to generate the datasets that satisfy the user-specified density level. Figure 4.7 shows the typical clusters in a dataset having a difficulty level of one. It can be seen that such design of the data distribution ensures that data are clearly divided into well-formed groups which makes it relatively easy for clustering algorithms to find the clusters. When evaluating clustering methods with these type of datasets, we are mostly concerned with how fast a certain method can identify the clusters in a large dataset.

Datasets with Difficulty Level Two

The datasets have long and thin clusters with straight or curved shapes. Like clusters in level one, clusters in a particular dataset are well separated. Figure 4.8 gives some of the example clusters in the datasets having a difficulty level of two. Based on the input parameters, linear equations and transformations of contraction, expansion, rotation and translation are performed on the basic dataset to create level-two

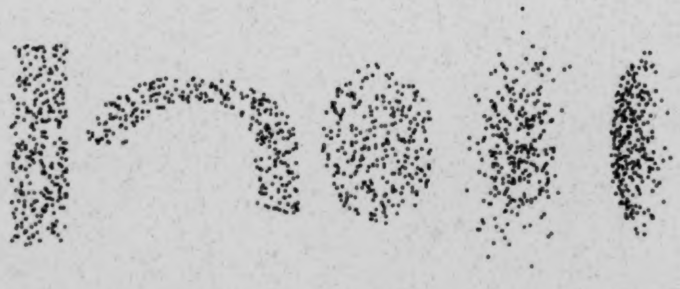


Figure 4.8: Difficulty level 2: each cluster contains 300 2D points

datasets. Although the clusters are at an easy level and are as intuitive as the first level ones, their enlongated shape can make some clustering methods fail in identifying them. For example, the algorithms k-means [51] and k-medoids [43] are most likely to split such a cluster into two or more groups as they favor spherical shaped clusters.

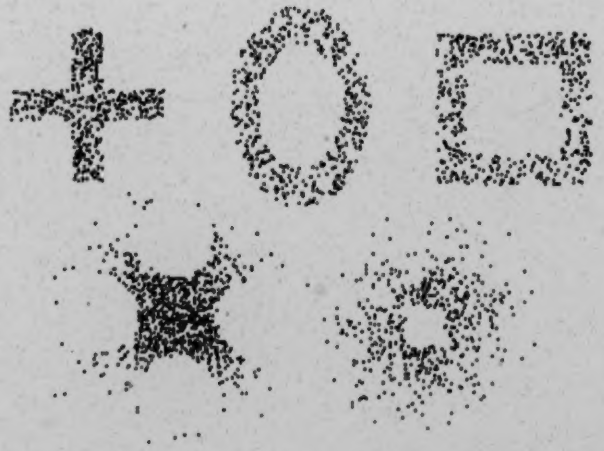


Figure 4.9: Difficulty level 3: each cluster contains 500 2D points

Datasets with Difficulty Level Three

The clusters in the dataset with difficulty level three have simple arbitrary shapes such as rings, crosses and stairs. Different clusters are clearly separated. Some typical clusters are given in Figure 4.9 in which the three clusters on top have uniform distributions and the two at the bottom have underlying normal distributions. In order to generate these datasets, linear transformations such as contraction, expansion, rotation and translation as well as linear equations and circle equations have been performed on the basic dataset of either uniform or normal distribution.

Compared to the first two level datasets that contain only basic convex clusters, the level-three datasets have clusters that do not necessarily have an object defined as the mean. For example, there is not any object that can be considered as the explicit mean for a ring shaped cluster. Consequently, the irregular shape of clusters will increase the difficulty in finding meaningful clusters for any clustering algorithm that uses a data point as the mean of a cluster.

Datasets with Difficulty Level Four

The clusters in the dataset with difficulty level four have arbitrary shapes with some obvious space inside a cluster. There are no nested clusters. To enrich the diversity of cluster shape, clusters with uniform distribution are specifically designed to be any of the twenty-six letters of the alphabet which are evenly positioned in a particular dataset. Each letter is treated as an individual cluster. The system provide two options for generating the required number of alphabet clusters. One is to randomly produce any of the letters. The other option allows the user to input letters of his own interest. The operations used to control the distribution and shape of the letters involve all the techniques previously mentioned including equations and transformations. Example clusters are shown in Figure 4.10 in which the let-

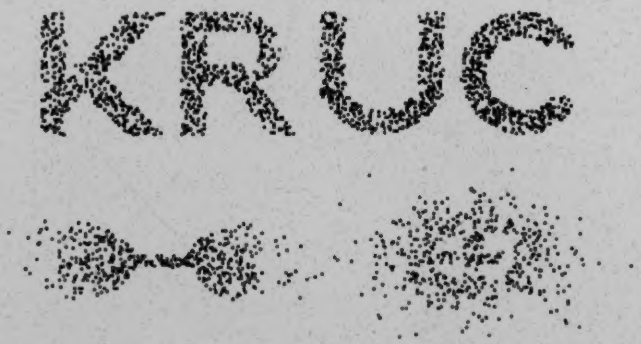


Figure 4.10: Difficulty level 4: each cluster contains 500 2D points

ters have uniform distributions and the other two clusters are generated by applying covariance matrix, linear equations and circle equations to standard normal distributions. Since the letters encompass a wide range of cluster shapes, it is hard for most clustering methods to find all the different letter-shaped clusters. Although density-based algorithms such as DBSCAN work well with datasets containing diverse cluster shape, they will fail in identifying some of the clusters if the densities between clusters are quite different.

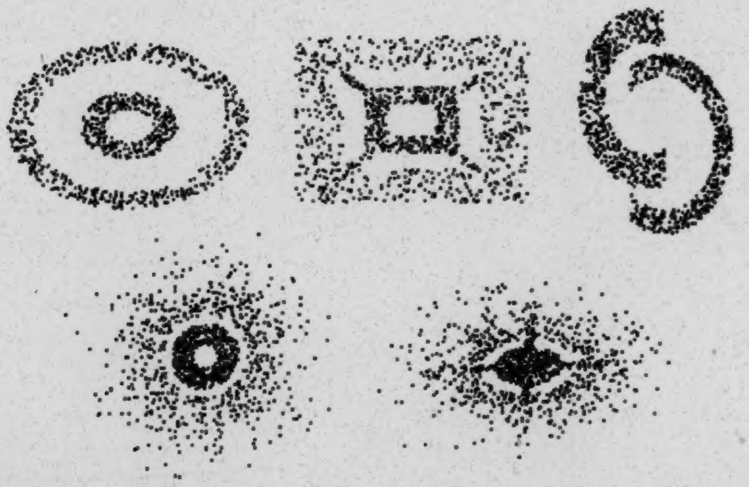


Figure 4.11: Difficulty level 5: each paired cluster contains 1,000 2D data points, of which 500 are assigned to each single cluster

Datasets with Difficulty Level Five

The datasets contain clusters within clusters or single clusters with irregular shapes. In the case of one cluster within the other cluster, the two clusters can either be clearly separated or they are connected with bridges of points, which can cause much trouble to many clustering algorithms in correctly identifying the clusters. Nested clusters also raise a question as to how to define a cluster: should we consider a nested cluster as one cluster or several clusters? Figure 4.11 displays some of the clusters in datasets having a difficulty level of five. In addition to the generation mechanism for creating clusters of the other levels, special attention is paid to positioning the nested clusters at this level by either controlling the mean value or applying transformations and equations.

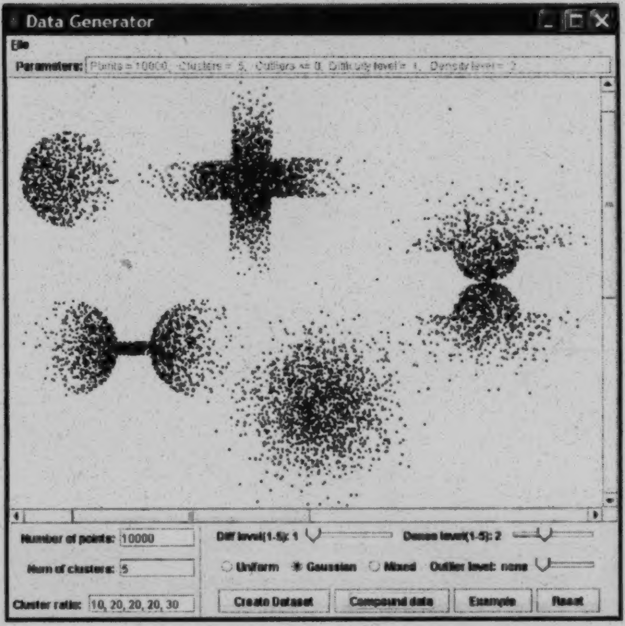
4.4.2 Generation of Outliers/Noise in a Dataset

As discussed in the previous chapters, outliers and noise are unavoidable in real life datasets collected from numerous application domains. The mechanism of adding outliers or noise is another important contribution of our synthetic data generation system. While there is no strict distinction between outliers and noise in most data analysis tasks, we will use outliers as a generic term in the following discussion.

It is well accepted that outliers in a dataset are not consistent with the rest of the data. This leads to the exploration of outlier detection based on the distance to a point's neighboring points. Many existing outlier detection methods use the neighborhood density of a point as a criterion to differentiating abnormalities from normalities. Points located in a less dense region are usually considered as outliers. While intuitive, such definition raises new issues: how do we specify the cutoff density value to guarantee real outliers and meaningful clusters? Should the points located in the outer layers of a normal distribution as shown in Figures 4.7 through 4.11 be marked as outliers? Or should all the data in a less dense cluster be treated as outliers?

Because the definition of outliers is subjective, the notions of outliers and inliers in a dataset are ambiguous in many situations. Data object being identified as outliers by one data analysis method could be legitimate inliers with the other method. Therefore, To produce outliers with respect to local and global clusters, our effort is focused on how to generate those data points that can be objectively identified as outliers by the existing outlier detection algorithms.

The method of generating outliers is similar to that of generating clusters. Standard distribution and linear transformation have been widely used. The distribution and density of outliers are determined by the system through the parameter: outlier level. The value of outlier level can be none, low and high and are specified by the user. Depending on the selected level, the number of outliers is a controlled percentage of the total number of points in a dataset. For example, the outliers account



Parameters: Points = 10000, Clusters = 5, Outliers = 0, Difficulty level = 1, Density level = 2

Number of points: 10000

Num of clusters: 5

Cluster ratio: 10, 20, 20, 20, 30

Diff level(1-5): 1

Dense level(1-5): 2

○ Uniform ● Gaussian ○ Mixed

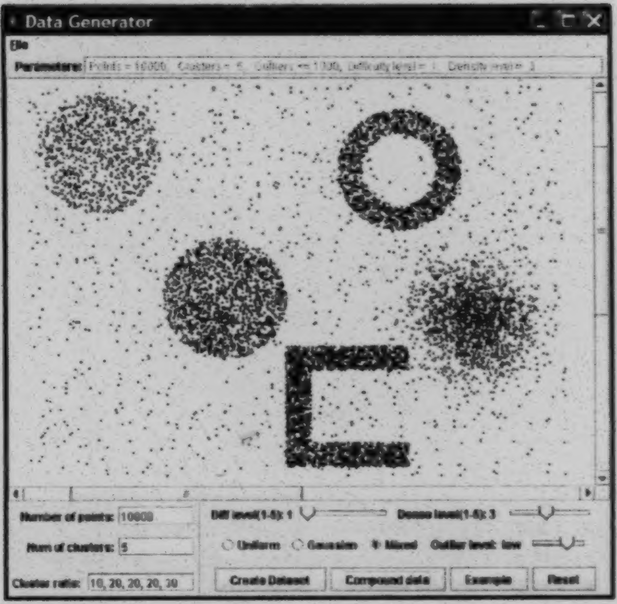
Outlier level: none

Figure 4.12: Outliers level none: outliers are those exterior points of a cluster

for 10% of the data when 90% of the data are in clusters. This ensures that the total number of points from the user input is preserved while outliers are being added. Next we will discuss the generation of the three level outliers. The examples used are all complex mixed datasets that contain clusters of different difficulty levels.

Outliers Level None

The name of "level none" is self-explaining. No outliers are intentionally added to a dataset. However, this does not necessarily mean that a set of generated data does not contain outliers. A dataset often consists of clusters with different distributions and densities. Depending on the definition of a specific outlier detection algorithms, data points in clusters of different difficulty levels as described before can be outliers. For example, a cluster itself can be considered as a collection of outliers if the size of the cluster is much smaller than those of other clusters or the data in the cluster are very sparsely distributed compared to the majority of the data. Most outlier detection algorithms would mark the exterior points in a normally distributed cluster as outliers. Such examples are demonstrated in Figure 4.12.



Parameters: Points = 10000, Clusters = 5, Outliers == 1000, Difficulty level = 1, Density level = 3

Number of points: 10000

Num of clusters: 5

Cluster ratio: 10, 20, 20, 20, 30

Diff level(1-5): 1

Dense level(1-5): 3

○ Uniform ○ Gaussian ● Mixed

Outlier level: low

Figure 4.13: Outliers level low: outliers are randomly distributed

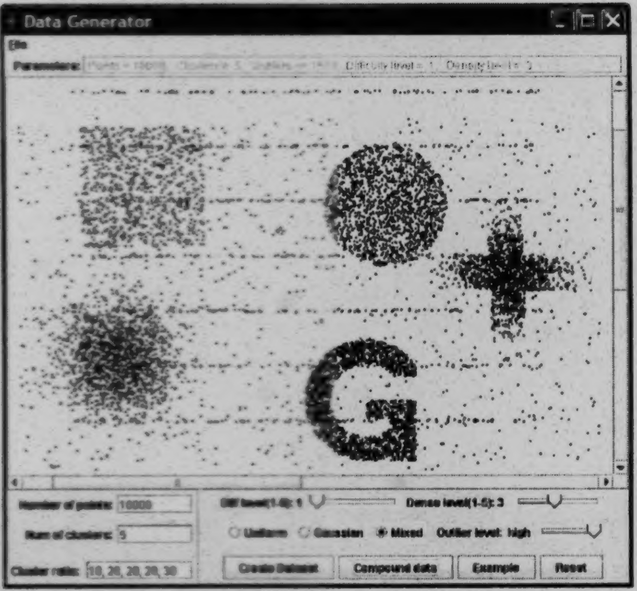
Outliers Level Low

This is the basic type of outliers. For a given dataset, the system randomly distributes a small percentage of the data in the whole data space. Figure 4.13 shows a dataset containing 4,000 points including outliers.

Outliers Level High

In addition to generating randomly distributed outliers, the data generator produces outliers of controlled shape and distribution. Since there is no universal agreement on what constitutes outliers, our intention is to provide a prototype of outlier distribution in a dataset. Figure 4.14 gives an example dataset containing 5,000 2D points in which outliers count up to 15% of the total data. Three types of data points can be classified as outliers in this dataset:

1. points that are located in a sparse neighborhood;
2. exterior points of the clusters that have normal distributions; and
3. points that form certain patterns, such as the lines, each of which has much



Number of points: 10000

Num of clusters: 5

Cluster ratio: 10, 20, 20, 20, 30

Diff level(1-6): 1

Dense level(1-5): 3

○ Uniform ○ Gaussian ● Mixed

Outlier level: high

Figure 4.14: Outliers level high: outliers are either randomly distributed or have simple patterns

less data than those major clusters. Some may not consider these points as outliers because they form a major pattern. Depending on the density of the lines, these points can be classified into either points in clusters or outliers. We will demonstrate this in section 4.5 with experimental results.

Many clustering and outlier analysis algorithms can easily identify the first two type of outliers that have sparse neighborhoods. But the third type of outliers can cause problems in the process of data clustering. For example, the density-based clustering algorithm DBSCAN has been well recognized as an effective method in finding clusters of arbitrary shapes as well as identifying and eliminating outliers. However, it may merge two or more clusters together when the lines or the so-called bridges of points join these clusters into a group.

4.5 Experiments and Evaluation

One of the most effective ways to evaluate the generated dataset is to visualize the data for human inspection. The GUI of the data generating system has been designed to serve this purpose. In addition to visual inspection, we test the performance of our system in two aspects:

- the efficiency of producing large datasets that satisfy user's requirement;
- the effectiveness of a benchmark instance generator for clustering analysis and outlier detection.

In this section, we report experiments and evaluation results of our synthetic data generation system.

4.5.1 Generating Very Large Datasets

We first test how the size of the generated datasets affects the execution time. For any dataset with size up to 1,000,000 points, the execution time for generating the data (excluding writing the data to a file) is less than three seconds. This is demonstrated in Figure 4.15, which is a plot of the execution time against the size of the generated datasets. It is observed that to generate a dataset containing less than 4,000,000 data points, the execution time is linear to the size of the dataset regardless of difficuty levels, density levels and outlier levels.

Since the difficulty level is the major factor that determines the distribution and shape of each cluster in a dataset, we also ran the program to show how the execution time is affected by the difficulty level. For each difficulty level (from 1 to 5), we input the same parameters which include data size, number of clusters, density and outlier levels so that the difference of data generating time is exclusively based on the change of difficulty levels. Despite the use of the same parameters, each dataset produced may contain clusters of different distributions, shapes and densities. In order to precisely show the execution time of generating a dataset, we ran the program at least five times for each difficulty level and then computed the average excution time. The plot in Figure 4.16 demonstrates that with the change of difficulty levels, there is little change of average execution time to generate a certain type of dataset.

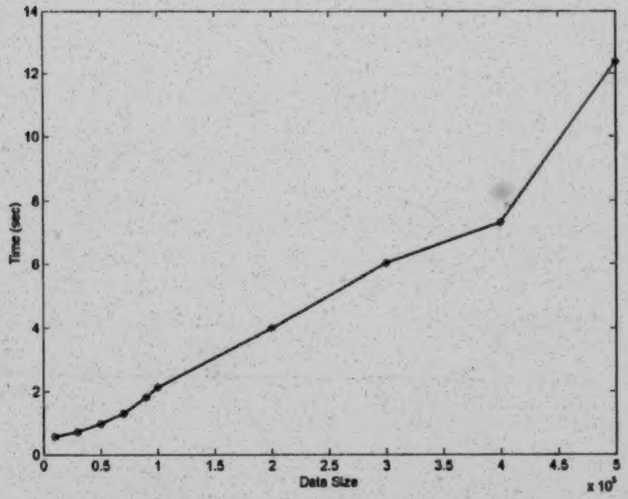


Figure 4.15: Each generated dataset has the following properties: number of clusters is 5; data distribution in a cluster is either uniform or normal; difficulty level ranges from 1 to 5, density level is 3, and noise level is low

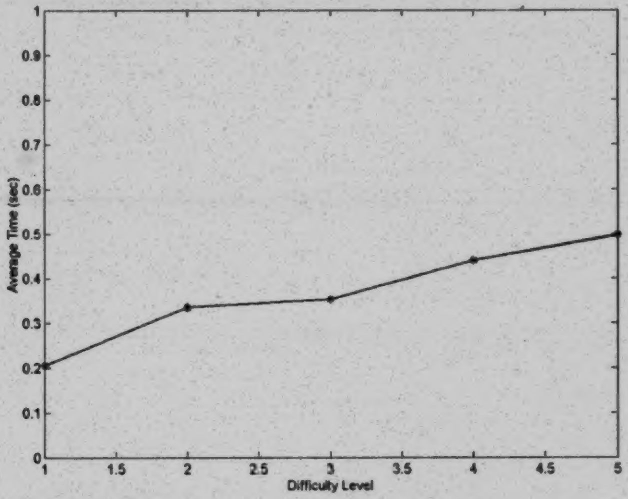


Figure 4.16: With each difficulty level, the system generates a dataset of 100,000 that contains both uniformly and normally distributed clusters.

4.5.2 Testing with Clustering and Outlier Analysis Algorithms

We generated six sets of two dimensional spatial data. Each dataset contains outliers as well as clusters that consist of either uniformly or normally distributed data. The details of the datasets are given in Table 4.1. Clusters in each of the first five datasets exhibit the typical cases of data distributions and shapes of a specific difficulty level. The sixth dataset, however, contains a mixture of clusters that are randomly generated from different difficulty levels. The sizes of the datasets are moderate for easy inspection and illustration.

Table 4.1: Detailed description of the parameters for the datasets

Dataset	**Size**	**Number of clusters**	**Difficulty level**	**Noise level**
dataset1	2,000	4	1	low
dataset2	2,000	4	2	low
dataset3	2,000	4	3	low
dataset4	2,000	5	4	high
dataset5	2,000	5	5	high
dataset6	10,000	7	mixed	high

Table 4.2: Description of the clustering algorithms

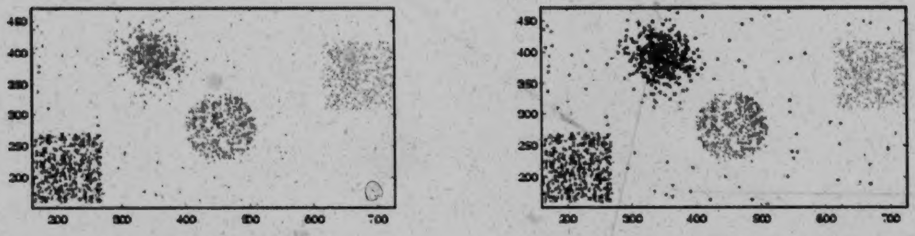
Algorithm	**classification**	**Parameters**
k-means	partition-based	k
DBSCAN	density-based	radius ϵ, $MinPts$
CURE	hierarchical	k, shrinking factor α, representative points t
CHAMELEON	hierarchical	$k-NN$, $MinSize$, k
WaveCluster	grid-based	resolution r, signal threshold τ
AutoClass	model-based	N/A

Using these datasets as benchmark instances, we conducted experimental evaluation upon six existing clustering algorithms: k-means [43], DBSCAN [20], CURE [27], CHAMELEON [42], WaveCluster [65] and AutoClass [15, 16]. The CURE code is kindly supplied by the Department of Computer Science and Engineering, University of Minnesota. The AutoClass is the public C version from [75]. The other four programs were locally implemented. Some basic characteristics of these clustering methods are generalized in Table 4.2.

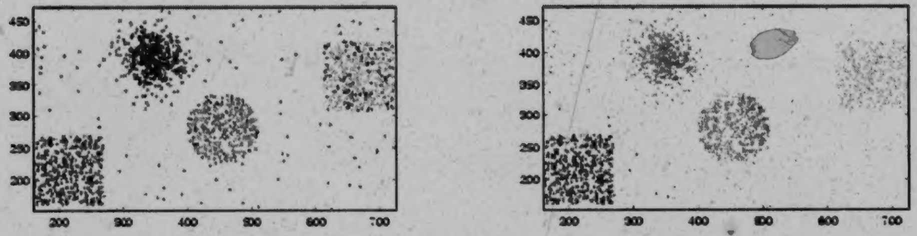
Our experiments proceed from *easy* datasets to *hard* datasets. The complexity of a dataset is defined by the difficulty levels as described in the previous section. Our intension is not to explore the different clustering mechanisms, instead, we aim to show experimentally how each of these six clustering algorithms performs with different datasets consisting of a diversity of clusters and difficulty levels. We show the clustering results on each dataset graphically to give a concrete idea of the clustering ability of different clustering methods. We assume that we have the specific domain knowledge of each dataset. When performing the experiment, such domain knowledge plays an important role in the selection of certain parameters, such as k, the number of clusters involved in some of the algorithms. To avoid the bias caused by inappropriate use of parameters for different algorithms, we also conduct many test-and-trials to select the set of parameters that lead to the best clustering results of the algorithm being tested.

In Figure 4.17 to 4.22, different colors have been employed to indicate discovered clusters in a dataset after the clustering process. Since some of these clustering methods, such as DBSCAN, CURE and WaveCluster, are able to identify outliers, red color is reserved to mark outliers in all the clustering results of the following figures. Figures 4.17 to 4.22 can be viewed in two ways:

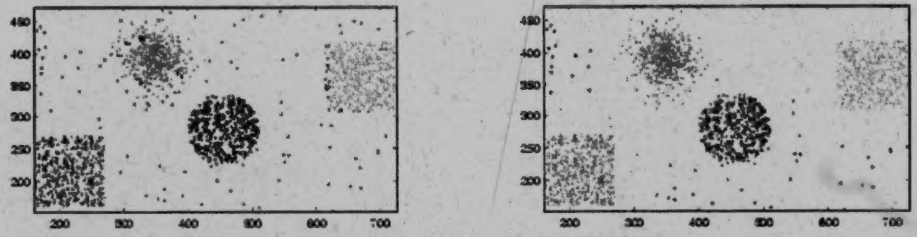
- Given a certain dataset, inspect the clustering abilities of different clustering algorithms, and
- For a certain clustering method, check its clustering results over different sets of data.



(a) k-means (b) DBSCAN

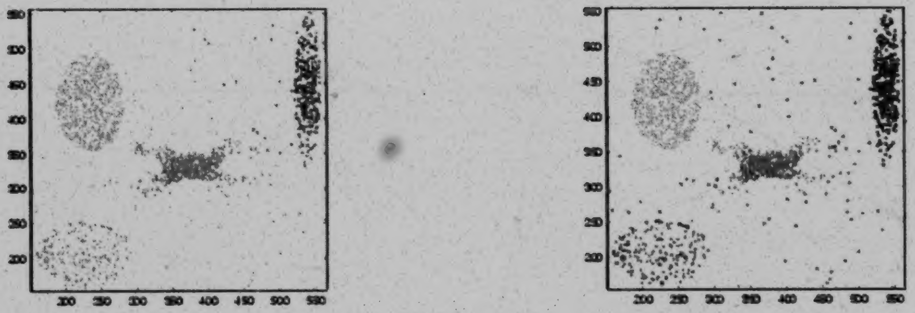


(c) CURE (d) CHAMELEON

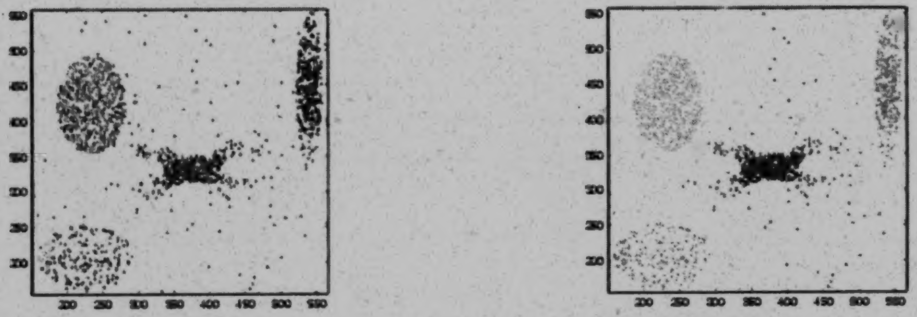


(e) WaveCluster (f) AutoClass

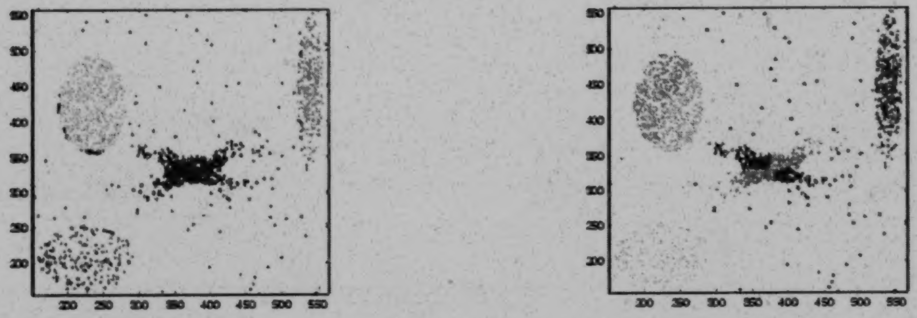
Figure 4.17: Clustering results on dataset 1. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



(a) k-means (b) DBSCAN

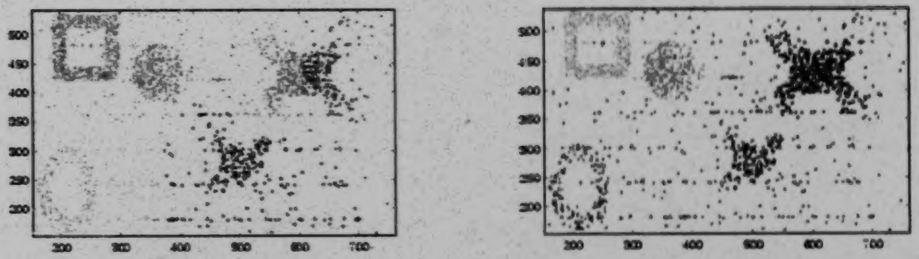


(c) CURE (d) CHAMELEON

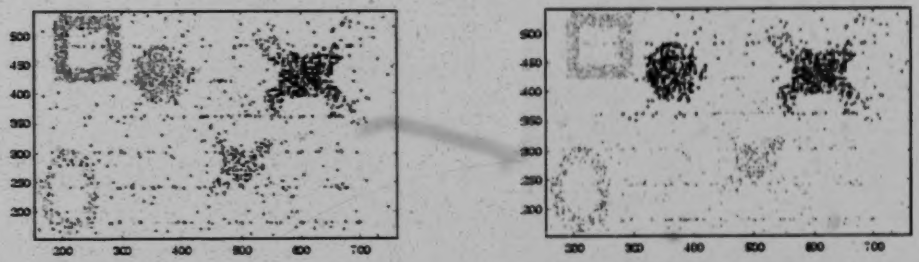


(e) WaveCluster (f) AutoClass

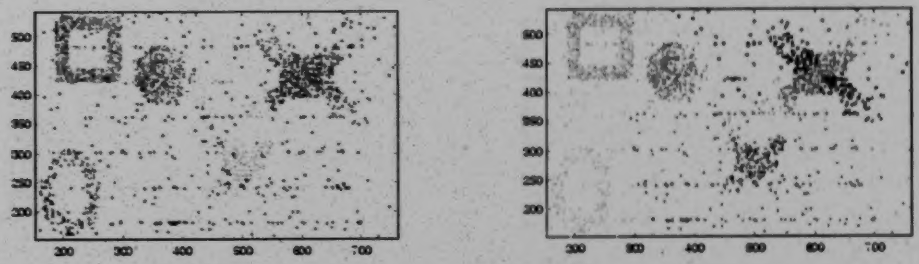
Figure 4.18: Clustering results on dataset 2. (a): k-mans with $k = 4$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 4$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 4$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



(a) k-means (b) DBSCAN

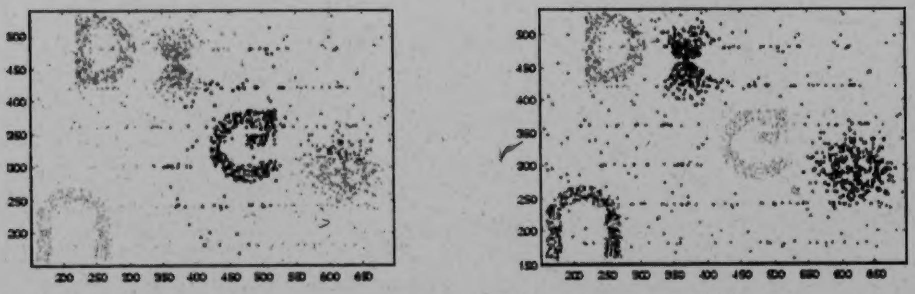


(c) CURE (d) CHAMELEON



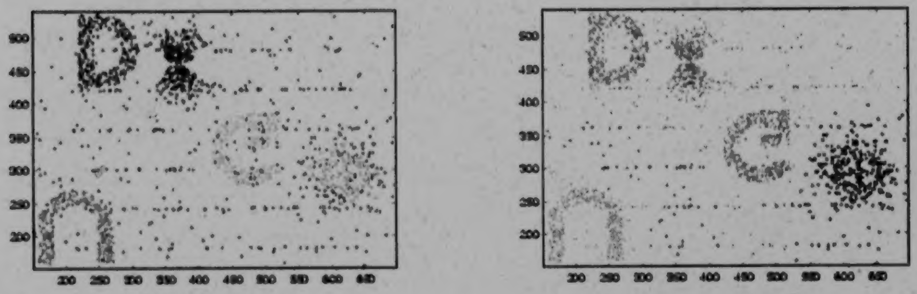
(e) WaveCluster (f) AutoClass

Figure 4.19: Clustering results on dataset 3. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



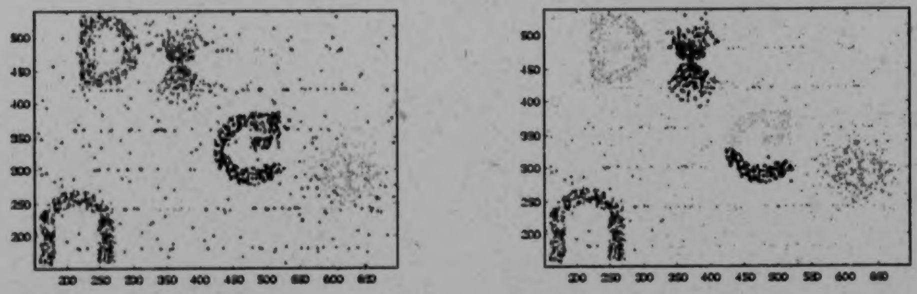
(a) k-means

(b) DBSCAN



(c) CURE

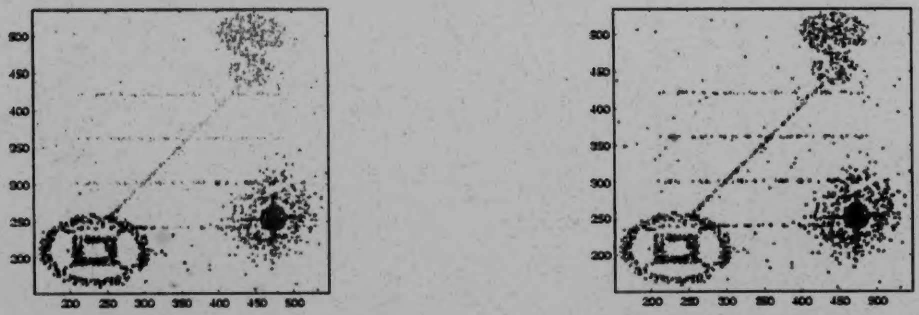
(d) CHAMELEON



(e) WaveCluster

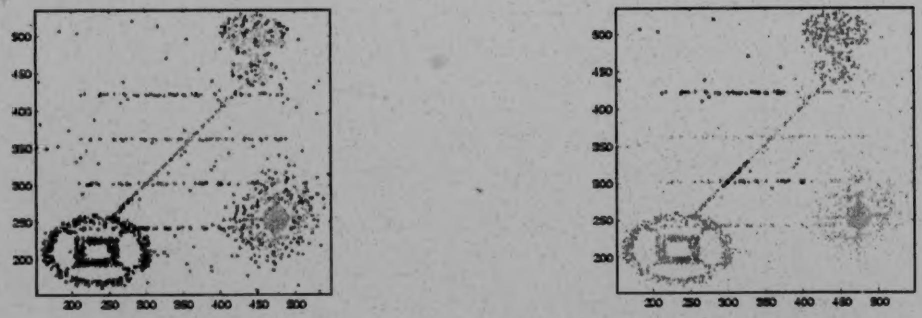
(f) AutoClass

Figure 4.20: Clustering results on dataset 4. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



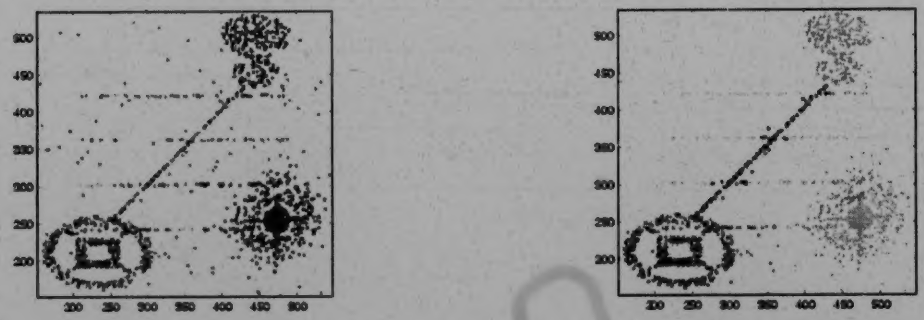
(a) k-means

(b) DBSCAN



(c) CURE

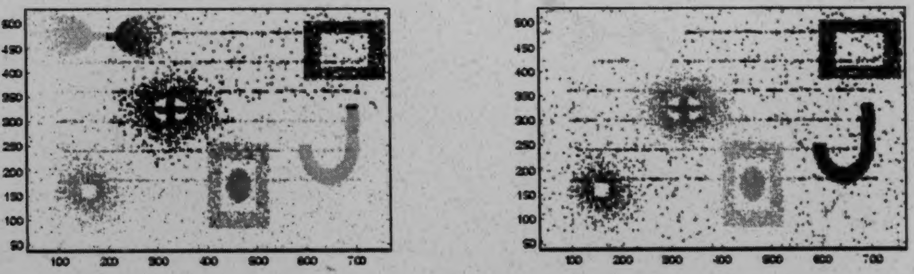
(d) CHAMELEON



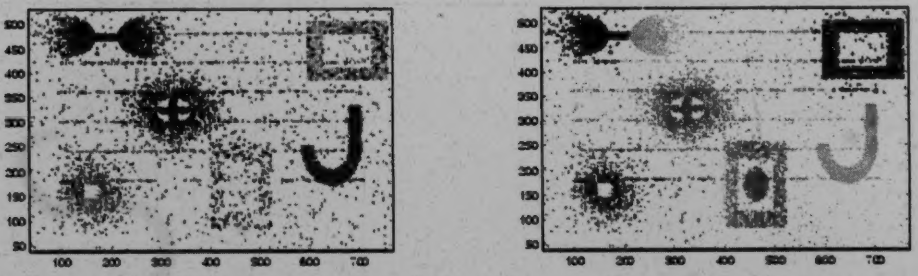
(e) WaveCluster

(f) AutoClass

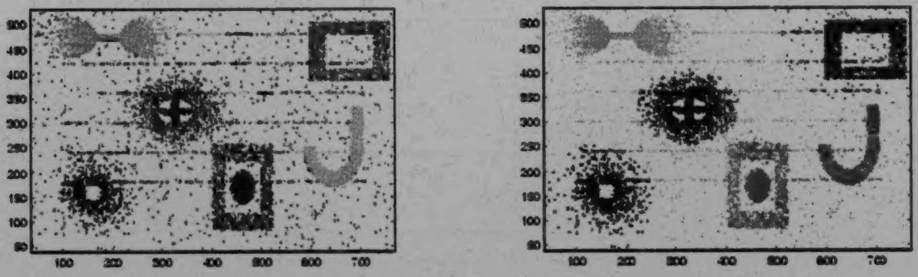
Figure 4.21: Clustering results on dataset 5. (a): k-mans with $k = 5$; (b): DBSCAN with $\epsilon = 15$ and $MinPts = 10$; (c): CURE with $k = 5$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 5$; (e): WaveCluster with $r = 5$ and $\tau = 0.2$; (f): AutoClass



(a) k-means (b) DBSCAN



(c) CURE (d) CHAMELEON



(e) WaveCluster (f) AutoClass

Figure 4.22: Clustering results on dataset 6. (a): k-means with $k = 7$; (b): DBSCAN with $\epsilon = 20$ and $MinPts = 30$; (c): CURE with $k = 7$, $\alpha = 0.3$, and $t = 10$; (d): CHAMELEON with $k - NN = 15$, MinSize=2.5%, and $k = 7$; (e): WaveCluster with $r = 4$ and $\tau = 0.2$; (f): AutoClass

The definition of clusters and outliers is often subjective. Meaningful clusters and real outliers should be considered in the context of application domains. Even with the synthetic datasets used in our experiments, it is sometimes not easy to mark clusters from clusters or to distinguish clusters from outliers. For example, in Figure 4.21, CURE and AutoClass treat the diagonal line pattern as a single cluster while other methods consider it either as part of another cluster or as outliers. Another example is the clustering results shown in Figure 4.22 obtained from dataset 6 , where the small oval and big rectangle (cluster in cluster) are grouped into one cluster by all the six clustering methods although they might well be considered as two clusters. Pros and cons of various clustering algorithms have been widely discussed in the literature, we evaluate these algorithms based on the quality of the clustering results on the given datasets.

Some interesting observation from the experiments can be generalized as follows.

1. K-means is well known for being able to quickly find spherical shaped clusters. Through the experiments on datasets of different levels, it is found that k-means can successfully identify irregular shaped clusters if the distances between clusters are big enough and the initial set of centroid have been well selected. Three major factors that mostly affect the clutering results of k-means are: (1) domain knowledge for the selection of parameter k; (2) initial location of the set of centroid; and (3) distribution of outliers.

2. Given the appropriate values for the two parameters: neighborhood radius ϵ and MinPts, DBSCAN achieves the best clustering results among the six algorithms. It can not only find arbitrary shaped clusters but can also detect most outliers. One intrinsic shortcoming of DBSCAN is that it may merge two or more clusters if there exist "bridges" of outliers joining clusters such as Figure 4.21 (b).

3. CURE is designed to not only find arbitrary-shaped clusters, but also identify outliers in a dataset. Our experiments indicate that CURE can successfully find meaningful clusters that have identical densities, but it also marks

many points that are located in the uniformly distributed clusters as outliers as demonstrated with all the six datasets. A big problem with CURE is that it might fail to find some real clusters when the densities of these clusters are relatively less than those of the other clusters in a dataset as shown in Figure 4.18 (c).

4. Like k-means, CHAMELEON can not handle outliers. Although it is extremely slow, it is more effective than k-means as it can find clusters of arbitrary shapes regardless of the distances between clusters.

5. In most cases, WaveCluster is effective in finding clusters and outliers in a dataset. Although the number of resulted clusters is often more than the actual number of clusters in a dataset as shown in Figure 4.19 (e), 4.20 (e), 4.21 (e) and 4.22 (e), major clusters usually stand out since they contains far more data objects than those small clusters. A further step to eliminate small clusters and mark the data objects in these clusters as outliers would surely improve the effectiveness of WaveCluster.

6. The most interesting clustering algorithm used in our experiments is AutoClass. It is an unsupervised Bayesian classification system that seeks a maximum posterior probability classification [75]. Such method has been widely used in statistics and machine learning. The uniqueness of AutoClass is that it can find data clusters that might not be identified as clusters by visual inspection. For example, the blue clusters in Figure 4.19 (f) and Figure 4.20 (f). This is due to the fact that AutoClass is able to find clusters that is maximally probable with respect to the underlying data model. Though not designed to identify outliers, AutoClass can generally classify outliers into one group even though they are usually separated by clusters.

4.6 Conclusion

In this chapter, we present a comprehensive approach to synthetic data generation for data analysis and demonstrate that the approach is very effective in generating testing datasets for clustering and outlier analysis algorithms. According to the user requirements, the approach systematically creates testing datasets based on different data distribution and transformation. Given the number of points and number of clusters, each dataset is controlled by data distribution, difficulty level, density level and outlier level. The difficulty level determines the overall characteristic (shape, position) of the clusters in a dataset, the density level mostly determines the size and density of each cluster. The generated datasets contain clusters of two standard distributions: the uniform distribution and/or the normal distribution. While the synthetic data generation system is effective in generating two-dimensional testing datasets to satisfy user's requirement, it is also efficient in generating very large dataset with arbitrary shaped clusters.

Chapter 5

Conclusions and Future Work

This thesis addresses outlier analysis and detection in the area of data mining. We investigate different techniques and methods, discuss recent work and existing problems in outlier mining. A novel approach to outlier detection is presented. The proposed method uses the relative degree of density with respect to a set of reference points to approximate the degree of density defined in terms of the k nearest neighbors of a data point. Candidate outliers are ranked based on the reference-based outlier score (ROS) that has been assigned to each data point. The worst case execution time of our algorithm is $O(Rn \log n)$, where n is the size of the dataset and R is the number of reference points. Detailed analysis and experiments show that our method can quickly find meaningful outliers in both synthetic and real world datasets and are highly scalable to very large dataset.

Synthetic data generation is an interesting topic in data mining. In many research areas, benchmark datasets are essential in evaluating the quality of a proposed technique. Methods of generating datasets for different purposes can be quite different. Our work concentrates on the generation of test instances for clustering and outlier analysis algorithms. We presented a distribution-based and transformation-based approach to synthetic data generation. Based on this approach, we designed and implemented a generic framework for synthetic data generation. It can dynamically generate datasets of different probability distributions with various difficulty levels in terms of clusters and outliers. We test our data generator in two aspects: generating various large datasets and using the generated datasets to test the existing clustering algorithms. Experiment results demonstrate that the proposed approach is very efficient in generating very large datasets of different types. The generated datasets can be used as benchmarks to test and compare different clustering and outlier analysis algorithms. In addition, most of the example datasets used in this thesis are generated by our synthetic data generator. To our knowledge, our system is probably the first synthetic data generation system that systematically generates datasets for evaluating the clustering and outlier analysis algorithms.

Future Work

In addition to further improving the computational efficiency of the proposed reference-based outlier detection approach, one important direction of future work is the determination of reference points. As has been discussed in Chapter 3, the performance of our algorithm, efficiency and accuracy, is closely related to the number of reference points and the location of the reference points. In addition to the grid-based progressive approach to the selection of the reference points explored in Chapter 3, it is worthwhile to investigate the possibility of using other reference selection approaches that take into consideration the domain knowledge of the dataset to achieve a better tradeoff. Integrating the proposed approach with other methods of outlier detection in the literature is also an interesting line of research.

Our work on synthetic data generation concentrates on the generation of test instances for clustering and outlier analysis algorithms. There are still much room for improving the current data generating system. (1) We plan to redesign the interface to visualize not only 2D but also 3D data. (2) The size of a cluster is controlled by the density level, which ensures that the number of points in a cluster is fixed, but also poses a problem, i.e., similarly shaped clusters with a specific density have basically the same size. Finding a better way to address this problem can produce various sized clusters with the same density in a dataset. (3) In another more or less theoretical direction, it would be interesting to discuss the meaning of the difficulty of the datasets.

Bibliography

[1] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.

[2] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2001.

[3] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM-SIGMOD Int. Conf. Management of Data (SIGMOD)*, pages 94–105, 1998.

[4] F. J. Anscombe. Rejection of outliers. *Technometrics*, 2, 1960.

[5] A. Arning, R. Agrawal, and P. Raghavan. A linear method for deviation detection in large databases. In *Proceedings of the International Conference on Knowledge Discovery and Data Mining*, pages 164–169, 1996.

[6] V. Barnett and T. Lewis. *Outliers in Statistical Data*. John Wiley & Sons, 1994.

[7] S. Bay. Orca: A program for mining distance-based outliers. Internet page. `http://www.isle.org/~sbay/software/orca/`.

[8] S. Bay and M. Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.

[9] J. L. Bentley. Multidimensional binary search trees used for associative searching. *CACM*, 18(9):509–517, 1975.

[10] S. Berchtold, D. Keim, and H.-P. Kreigel. The X-tree: an index structure for high-dimensional data. In *Proceedings of the 22nd International Conference on Very Large Databases*, pages 28–39, 1996.

[11] R. J. Bolton and D. J. Hand. Statistical fraud detection: A review. *Statistical Science*, 17(3):235–255, 2002.

[12] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. OPTICS-OF: Identifying local outliers. In *Proc. 3rd European Conf. on Principles and Practice of Knowledge Discovery in Databases (PKDD'99)*, pages 262–270, September 1999.

[13] M. Breunig, H.-P. Kriegel, R. Ng, and J. Sander. LOF: Identifying density-based local outliers. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 93–104, May 2000.

[14] S. Chakrabarti, S. Sarawagi, and B. Dom. Mining surprising patterns using temporal description length. In *Proceedings of the 24th VLDB Conference*, pages 606–617, 1998.

[15] P. Cheeseman, J. Kelly, M. Self, J. Stutz, W. Taylor, and D. Freeman. Autoclass: A bayesian classification system. In *PProceedings of the Fifth International Conference on Machine Learning*, pages 54–56. Morgan Kaufmann Publishers, June 1988.

[16] P. Cheeseman and J. Stutz. Bayesian classification (autoclass): Theory and results. In U. Fayyad, G. Paitesky-Shapiro, P. Smyth, and R. Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 153–180. AAAI Press, 1995.

[17] Z. Chen, A. Fu, and J. Tang. On complementarity of cluster and outlier detection schemes. In *Proceedings of 5th International Conference on Data Warehousing and Knowledge Discovery, (DaWaK)*, pages 3–5, September 2003.

[18] J. Edvardsson. A survey on automatic test data generation. In *Proceedings of the Second Conference on Computer Science and Engineering in Linkping (CCSSE'99)*, October 1999.

[19] E. Eskin, A. Arnold, M. Prerau, L. Portnoy, and S. Stolfo. A geometric framework for unsupervised anormaly detection: Detecting intrusions in unlabeled data. In *Proc. Data Mining for Security Applications*, 2002.

[20] M. Ester, H-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996.

[21] V. Estivill-Castro and D. Wood. A survey of of adaptive sorting algorithms. *ACM Computing Surveys*, 24, 1992.

[22] U. M. Fayyad, S. G. Djorgovski, and N. Weir. Automating the analysis and cataloging of sky surveys. In U. M. Fayyad, G. Piatetsky-Shapiro, P. Smith, and R.Uthurusamy, editors, *Advances in Knowledge Discovery and Data Mining*, pages 471–493. MIT Press, 1996.

[23] A. Foss, W. Wang, and O. R. Zaïane. A non-parametric approach to web log analysis. In *Proc. of Workshop on Web Mining in First International SIAM Conference on Data Mining (SDM2001)*, pages 41–50, April 2001.

[24] A. Foss and O. R. Zaïane. A parameterless method for efficiently discovering clusters of arbitrary shape in large datasets. In *Proceedings of the IEEE International Conference on Data Mining (ICDM'2002)*, pages 179–186, December 2002.

[25] W. Frawley, G. Piatetsky-Shapiro, and C. Matheus. Knowledge discovery in databases: An overview. In G. Piatetsky-Shapiro and W. Frawley, editors, *Knowledge Discovery in Databases*, pages 1–27. AAAI/MIT Press, 1991.

[26] M. H. Goldwasser, D. S. Johnson, and C. C. McGeoch. *Data Structures, Near Neighbor Searches, and Methodology: Fifth and Sixth DIMACS Implementation Challenges*. American Mathematical Society, 2002.

[27] S. Guha, R. Rastogi, and K. Shim. CURE: An efficient clustering algorithm for large databases. In *In Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 73–84, 1998.

[28] S. Guha, R. Rastogi, and K. Shim. ROCK: a robust clustering algorithm for categorical attributes. In *Proceedings of the 15th International Conference on Data Engeneering (ICDE)*, pages 512–521, 1999.

[29] N. Gupta, A. Mathur, and M. L. Soffa. Automated test data generation using an iterative relaxation method. In *ACM SIGSOFT Foundations of Software Engineering*, pages 231–244, November 1998.

[30] R. Guttmann. A dynamic index structure for spatial searching. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 47–57, 1984.

[31] J. Han and M. Kamber. *Data Mining: concepts and Techniques*. Morgan Kaufmann Publishers, 2001.

[32] D. Hand, H. Mannila, and P.Smyth. *Principles of Data Mining*. The MIT Press, 2001.

[33] D Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.

[34] S. Hawkins, H. X. He, G. J. Williams, and R. A. Baxter. Outlier detecting using replicator neural networks. In *Proc. DaWaK*, 2002.

[35] R. Hecht-Nielsen. Replicator neural networks for universal optimal source codeing. *Science*, 269, 1995.

[36] HMC. Geometry of linear transformations of the plane. Internet page. `http://www.math.hmc.edu/calculus/tutorials/`.

[37] V. J. Hodge and J. Austin. A survey of outlier detection methodologies. *Artificial Intelligence Review*, 22, 2004.

[38] IBM. Intelligent information systems. Internet page. `http://www.almaden.ibm.com/software/quest/resources/`.

[39] P. Indyk. Nearest neighbors in high-dimensional spaces. In J. E. Goodman and J. O'Rourke, editors, *to appear in Handbook of Discrete and Computational Geometry*. CRC Press LLC, 2004.

[40] W. Jin, A. K. H. Tung, and J. Han. Mining top-n local outliers in large databases. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, August 2001.

[41] J. J. Jung and G.-S. Jo. Semantic outlier analysis for sessionizing web logs. In *EWMF'03 Workshop*, 2003.

[42] G. Karyapis, E.-H. Han, and V. Kumar. CHAMELEON: A hierarchical clustering algorithms using dynamic modeling. *IEEE Computer*, 32(8):68–75, 1999.

[43] L. Kaufman and P. J. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley & Sons, 1990.

[44] E. M. Knorr. *Outliers and Data Mining: Finding Exceptions in Data*. PhD thesis, University of British Columbia, April 2002.

[45] E. M. Knorr and R. T. Ng. Algorithms for mining distance-based outliers in large datasets. In *Proceedings of the 24th VLDB Conference*, pages 392–403, August 24-27 1998.

[46] E. M. Knorr and R. T. Ng. Finding intensional knowledge of distance-based outliers. In *Proceedings of the 25th VLDB Conference*, 1999.

[47] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *The VLDB Journal*, 8(3):237–253, 2000.

[48] B. Korel. Automated software test data generation. *IEEE transactions on software engineering*, 16(8), 1990.

[49] A. Lazarevic, L. Ertoz, V. Kumar, A. Ozgur, and J. Srivastava. A comparative study of anomaly detection schemes in network intrusion detection. In *Third SIAM Conference on Data Mining*, May 2003.

[50] J. K. Ludedeman and S. M. Lukawecki. *Elementary linear algebra*. West Publishing Company, 1986.

[51] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. University of California Press, 1967.

[52] Mathworld. Box-muller transformation. Internet page. `http://mathworld.wolfram.com/`.

[53] R. Ng and J. Han. Efficient and effective clustering method for spatial data mining. In *Proceedings. of 1994 Int'l Conf. on Very Large Data Bases (VLDB'94)*, pages 144–155, September 1994.

[54] NHL.com. Nhl.com. Internet page. `http://www.nhl.com`.

[55] J. W. Osborne and A. Overbay. The power of outliers (and why researchers should always check for them). *Practical Assessment, Research, and Evaluation*, 9(6), 2004.

[56] Y. Pei and O. Zaiane. http://www.cs.ualberta.ca/ yaling/cluster/. Internet page. `http://www.cs.ualberta.ca/~yaling/Cluster/`.

[57] Y. Pei and O. Zaiane. A synthetic data generator for clustering and outlier analysis. Technical report, TR06-15, Department of Computing Science, University of Alberta, June 2006.

[58] F. Preparata and M. Shamos. *Computational Geometry: an Intrduction*. Springer-Verlag, 1988.

[59] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. In *Proceedings of the ACM SIGMOD Conference*, pages 427–438, June 2000.

[60] D. Ren, B. Wang, and W. Perrizo. Rdf: A density-based outlier detection method using vertical data representation. In *Proc. of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 503–506, 2004.

[61] S. Ross. *A first course in probability*. Prentice Hall, 1997.

[62] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2nd edition, 2003.

[63] I. Ruts and P. Rousseeuw. Computing depth contours of bivariate point clouds. *Computational Statistics and Data Analysis*, 23, 1996.

[64] H. Samet. *The Design and Analysis of Spatial Data Structures*. Addison-Wesley, 1990.

[65] G. Sheikholeslami, S. Chatterjee, and A. Zhang. WaveCluster: A multi-resolution clustering approach for very large spatial databases. In *Proceedings of 24th International Conference on Very Large Data Bases*, August 1998.

[66] S. Shekhar, C. T. Lu, and P. Zhang. Detecting graph-based spatial outliers. *International Journal of Intelligent Data Analysis (IDA)*, 6(5):451–468, 2002.

[67] S. Shekhar, C.T. Lu, and P. Zhang. Detecting graph-based spatial outliers: Algorithms and applications. In *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining*, 2001.

[68] StatCrunch. Statistical software for data analysis on the web. `http://www.statcrunch.com/`.

[69] Statlets. Statpoint internet statistical computing center. Internet page. `http://www.statlets.com/`.

[70] P. Sun and S. Chawla. On local spatial outliers. In *Proc. of the Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 209–216, 2004.

[71] J. Tang, Z. Chen, A. Fu, and D. W. Cheung. Enhancing effectiveness of outlier detections for low density patterns. In *Proc. 6th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD-02)*, May 2002.

[72] Y. Theodoridis and M. Nascimento. Generating spatiotemporal datasets on the www. *ACM SIGMOD Record*, 29(3), September 2000.

[73] A. Turner. Density data generation for spatial data mining applications. In *Proceedings of the 5th International Conference on GeoComputation*, August 2000.

[74] VBRC. Viral bioinformatics resource center. Internet page. `http://athena.bioc.uvic.ca/techDoc/`.

[75] T. Will. NASA ames research center: The autoclass project. Internet page. `http://ic.arc.nasa.gov/ic/projects/bayes-group/`.

[76] G. Williams, R. Baxter, H. He, S. Hawkins, and L. Gu. A comparative study of RNN for outlier detection in data mining. In *Proc. ICDM*, Dec. 2002.

[77] Y. Yu, D. Ganesan, L. Girod, D. Estrin, and R. Govindan. Synthetic data geneartion to support irregular sampling in sensor networks. In *Geo Sensor networks*, October 2003.

[78] O. R. Zaiane, A. Foss, C.-H. Lee, and W. Wang. On data clustering analysis: Scalability, constraints and validation. In *Proc. of the Sixth Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD'02)*, pages 28–39, May 2002.

[79] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 103–114, Montreal, Canada, June 1996.

Appendix A

A GUI-based Web Application for Data Clustering and Outlier Detection

Data clustering has attracted the attention of many researchers in different disciplines. It is an important and useful technique in data analysis. A large number of clustering algorithms have been put forward and investigated. Many of the existing clustering approaches are able to identify outliers/noise as well as clusters. The web portal serves as a place for both research and educational learning in the area of data mining, specifically, data clustering analysis. It is an on-line source to provide data mining researchers with implementations of some well-known clustering algorithms and a virtual lab to compare clustering results obtained from different approaches on various datasets.

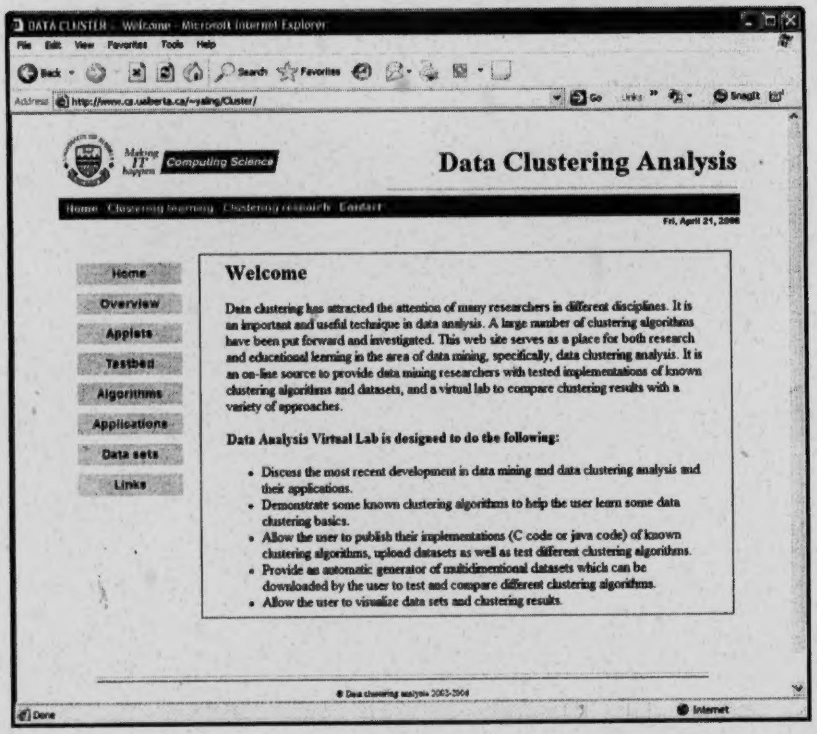
A.1 Goals

The Web-based data analysis system [1] is designed for two types of users in the area of data mining: beginners to learn the basics of existing clustering methods and researchers to conduct experiments on clustering analysis. It aims to do the following tasks:

- Discuss the concept and recent development in data clustering analysis and their applications.

---

[1] http://www.cs.ualberta.ca/ yaling/Cluster


DATA CLUSTER - Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.ualberta.ca/~yaling/Cluster/
Go Links Snagit
Making IT happen
Computing Science
Home Clustering learning Clustering research Contact
Fri, April 21, 2006
Home
Overview
Applets
Testbed
Algorithms
Applications
Data sets
Links
© Data clustering analysis 2003-2004
Done
Internet

Data Clustering Analysis

Welcome

Data clustering has attracted the attention of many researchers in different disciplines. It is an important and useful technique in data analysis. A large number of clustering algorithms have been put forward and investigated. This web site serves as a place for both research and educational learning in the area of data mining, specifically, data clustering analysis. It is an on-line source to provide data mining researchers with tested implementations of known clustering algorithms and datasets, and a virtual lab to compare clustering results with a variety of approaches.

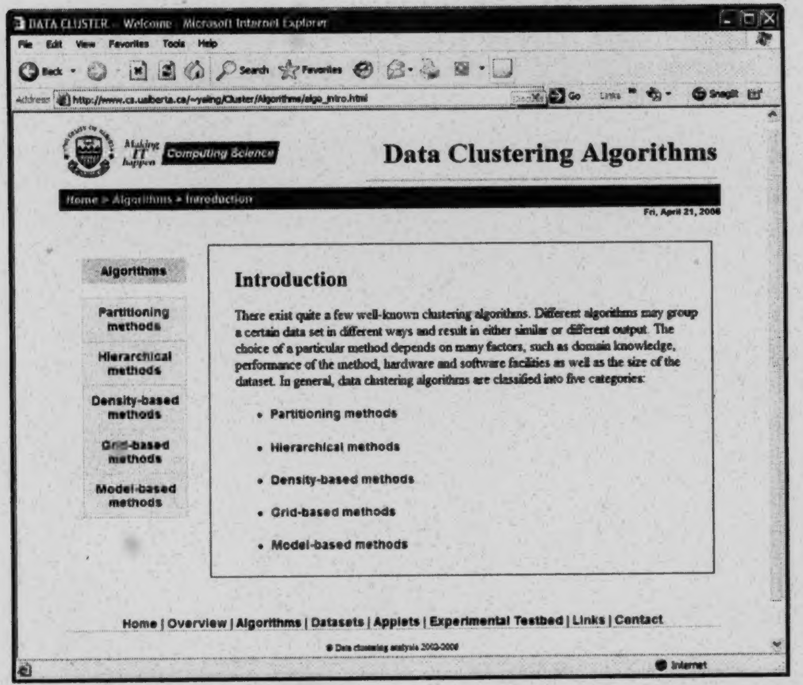
Data Analysis Virtual Lab is designed to do the following:

- Discuss the most recent development in data mining and data clustering analysis and their applications.
- Demonstrate some known clustering algorithms to help the user learn some data clustering basics.
- Allow the user to publish their implementations (C code or java code) of known clustering algorithms, upload datasets as well as test different clustering algorithms.
- Provide an automatic generator of multidimentional datasets which can be downloaded by the user to test and compare different clustering algorithms.
- Allow the user to visualize data sets and clustering results.

Figure A.1: Data clustering analysis - welcome

- Demonstrate some well-known clustering algorithms visually to help the user learn some data clustering basics.
- Provide a GUI-based Web interface for various clustering algorithms which are originally implemented in standalone applications.
- Allow the user to run clustering algorithms without logging into any remote machines. The user may upload their own datasets and test these datasets with different clustering algorithms.
- Visualize the clustering results obtained from different clustering algorithms so that the user can view the output as well as compare and evaluate different algorithms.

The screen shot of the welcome page is shown in Figure A.1.


DATA CLUSTER - Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address http://www.cs.ualberta.ca/~yaling/Cluster/Algorithms/algo_intro.html
Go
Links
SnagIt
Making IT happen
Computing Science
Internet

Data Clustering Algorithms

Home > Algorithms > Introduction

Fri, April 21, 2006

Algorithms

Partitioning methods

Hierarchical methods

Density-based methods

Grid-based methods

Model-based methods

Introduction

There exist quite a few well-known clustering algorithms. Different algorithms may group a certain data set in different ways and result in either similar or different output. The choice of a particular method depends on many factors, such as domain knowledge, performance of the method, hardware and software facilities as well as the size of the dataset. In general, data clustering algorithms are classified into five categories:

- Partitioning methods
- Hierarchical methods
- Density-based methods
- Grid-based methods
- Model-based methods

Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact

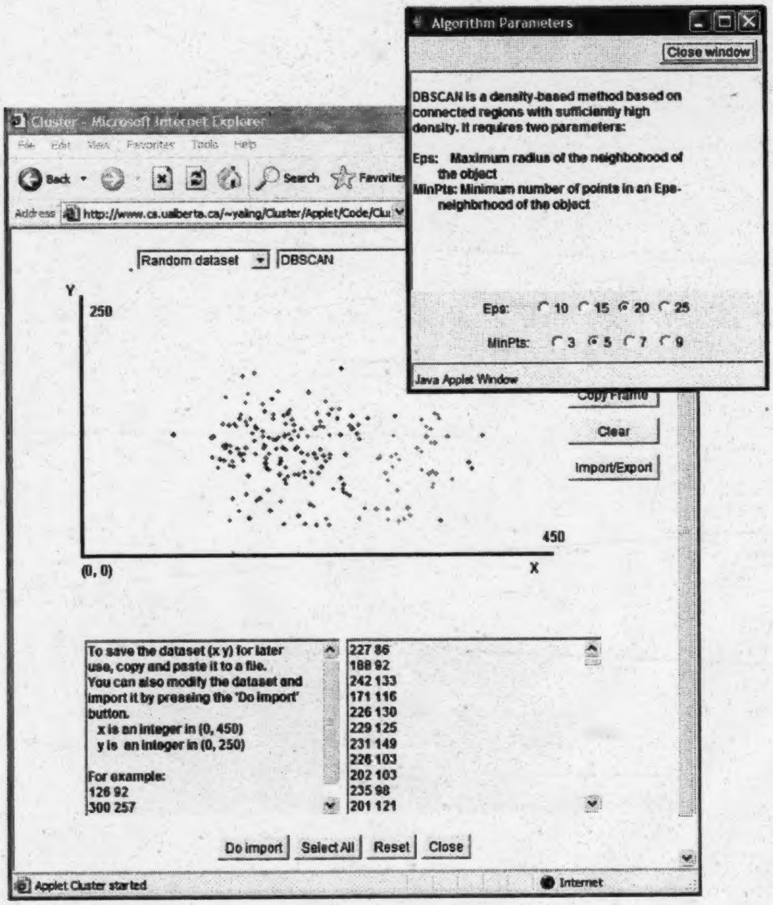
© Data clustering analysis 2002-2006

Figure A.2: Data clustering analysis - algorithms

A.2 A Virtual Lab for Learning Clustering methods

As shown in Figure A.2, we adopt the well known classification of clustering techniques: Partitioning methods, hierarchical methods, density-based methods, grid-based methods and model-based methods. Different algorithms may group a certain data set in different ways and result in either similar or different output. The choice of a particular method depends on many factors, such as domain knowledge, performance of the method, hardware and software facilities as well as the size of the dataset. We provide in this Web site brief reviews and links to the original papers of each clustering method. Interested readers are referred to the Web site for details of various clustering techniques and their applications.

In addition to the discussion of each method, we also provide a Java applet to demonstrate visually the working mechanisms of the well-known clustering methods: Kmeans, Kmedoids, DBSCAN and CLIQUE. Explanation of the algorithms and the user manual for running the applet are provided along with the applet. Fig-


Algorithm Parameters
Close window
Cluster - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.ualberta.ca/~yaling/Cluster/Applet/Code/Clu
Random dataset
DBSCAN
Y
250
450
(0, 0)
X
Copy Frame
Clear
Import/Export
Do import
Select All
Reset
Close
Java Applet Window
Applet Cluster started
Internet

DBSCAN is a density-based method based on connected regions with sufficiently high density. It requires two parameters:

Eps: Maximum radius of the neighbohood of the object
MinPts: Minimum number of points in an Eps-neighbrhood of the object

Eps: ○ 10 ○ 15 ● 20 ○ 25

MinPts: ○ 3 ● 5 ○ 7 ○ 9

To save the dataset (x y) for later use, copy and paste it to a file.
You can also modify the dataset and import it by pressing the 'Do import' button.
x is an integer in (0, 450)
y is an integer in (0, 250)

For example:
126 92
300 257

```
227 86
188 92
242 133
171 116
226 130
229 125
231 149
226 103
202 103
235 98
201 121
```

Figure A.3: Data clustering analysis - applet

ure A.3 displays a screen shot of the applet showing the running result of DBSCAN with $Eps = 20$ and $MinPts = 5$ on a set of randomly generated data. The applet has the following functionalities:

- Allow the user to select an algorithm and set the parameters specific to the selected algorithm from a separated window.
- Allow the user to import his own datasets as well as using the default datasets provided by the system to run different algorithms.
- Allow the user to export and save the data.
- Allow the user to choose the running speed.
- Copy and save the running result in a separate window to facilitate comparison of different algorithms on a given dataset.

A.3 An Interactive Web-based Testbed for Clustering Analysis

We migrate the following clustering algorithms: Kmeans, DBSCAN, CURE, ROCK, CHAMELEON, CLIQUE, WaveCluster and AutoClass to an interactive Web-based application system. Based on the structural analysis and program understanding of the existing command line implementation of these clustering algorithms, we decide to keep the original source code for the clustering task, and write a wrapper for Web enabling purpose. We then further extend the system by displaying the clustering results graphically on Web browser at the client side. Without logging into any remote machines, users are able to run the algorithms directly from the server side once the applications are Web enabled.

System Design and Tools

The original implementation of the clustering algorithms are either written in C or Java. They are command line based standalone applications. To run each program,

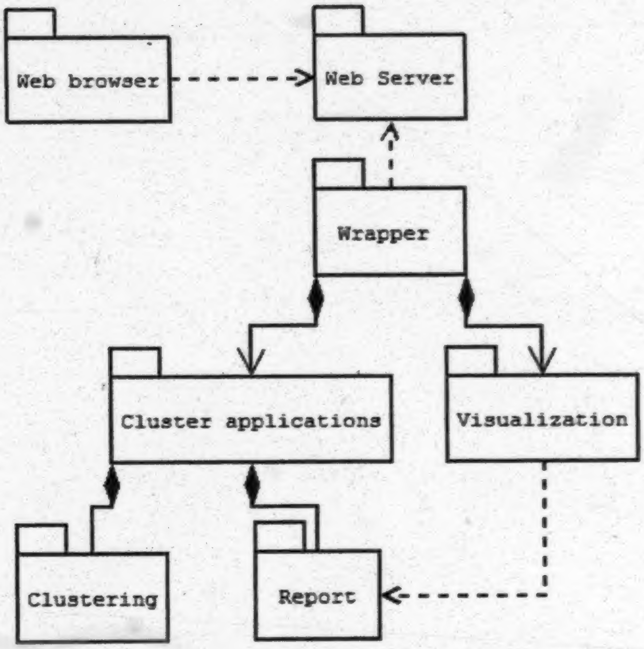


Figure A.4: UML of the Web-based clustering system

a user has to type in the command line the necessary parameters which require significant application domain knowledge. To enable the existing implementation on Web, the first question in our design phase is: What components of the system will be exposed to the Web? In our interactive system, we address only the accessibility of data and presentation.

Although we try to wrap the standalone applications as they are, we found that minor modification of the original implementation is necessary due to the command line interaction between users and the running program. As we intend to encapsulate the internal mechanism with a wrapper, we have to disable any user intervention while the program is running.

HTML forms are used for submitting user queries. After running an algorithm on the server side, the clustering result is to be displayed on the client side. Therefore, the target system provides support for visualizing the clustering results. Figure A.4 presents the UML of the infrastructure of the Web-based interactive system.

The implementation in building the clustering testbed and migrating the existing applications into Web-based systems involves mostly four programming languages:

- PHP - a server-side scripting language for displaying and submitting user input as well as wrapping the original application
- Javascript - a client-side scripting language for program parameter selection
- Python - an object-oriented programming language used for parsing parameters, read and write to files
- Java Applet - visualizing two-dimensional clustering results on Web browser

System Functionalities

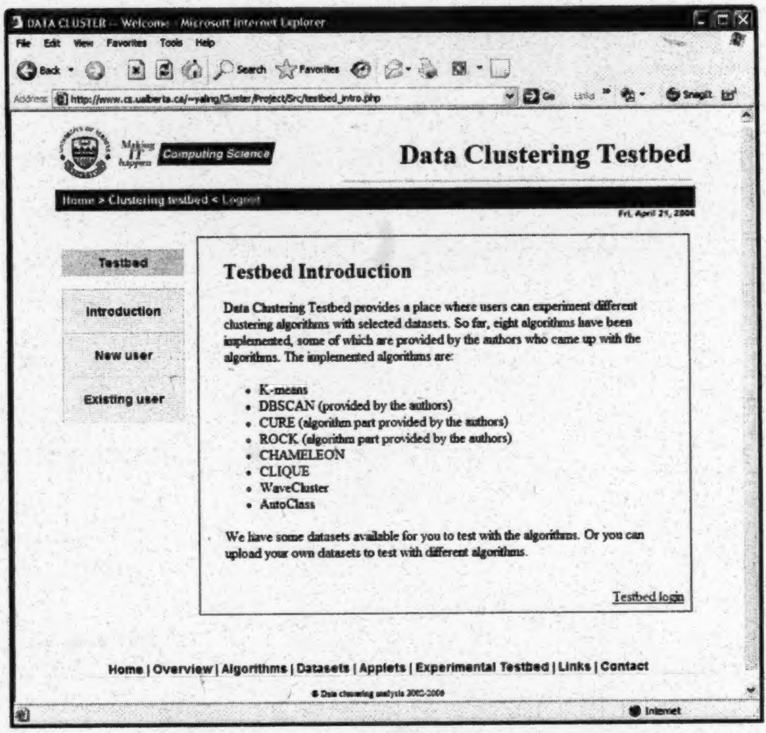
We explain how the clustering testbed works by walking through the Web pages. Figure A.5 through A.11 show the steps from logging into the system to getting the clustering result. In general, there are five major steps in running the application to obtain the clustering result.

1. Select an algorithm as shown in Figure A.8.
2. Select a dataset and enter the parameter values in the text boxes that are provided dynamically based on the selected algorithm as shown in Figure A.9.
3. Run the algorithm and check the parameters and clustering result.
4. Visualize the cluster result.

Many users without domain knowledge may want to run the clustering algorithms to evaluate the clustering results. They do not care about the internal mechanism as to how the system works. Even those users who are expert in clustering analysis may run the system only for the purpose of evaluating their algorithms and results. In both cases, it is not reasonable to ask users to specify all the parameters associated to the selected algorithm in order to run the program. To simplify the parameter selection task, we provide an extra text field to allow users to input an XML file name to save job history. When the user wants to run the same algorithm with the same setting for different datasets, the user does not need to type in these parameters repeatedly. Instead, by selecting the previously saved XML file, the system will automatically parse the parameters and run the program as desired. The benefit of saving job history can be more obvious when the system gets larger. In short, our Web-based interface eases users from checking the domain specific information.

Notice that the textual presentation of the clustering results provides no clue about how the data are distributed. For a dataset with even more than a hundred data points, it is almost impossible to evaluate the clustering results based only on the text information. As part of the testbed evaluation process, we decide to extend the original system by visualizing the clustering results, restricted to two-dimensional data. Java applet is used for this task so that it can be easily integrated with the Web-based system.

The user interface for visualizing the results as shown in Figure A.11 is designed to have a simple and clear layout. Two-dimensional data are represented as points on Web browser. To distinguish among resulting clusters, different colors are used for each cluster to help users understand the clustering results. Moreover, the applet also displays the relevant parameters such as algorithm name, total number of data and number of final clusters.


DATA CLUSTER -- Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/testbed_intro.php
Computing Science
Home > Clustering testbed < Logout
Fri, April 21, 2006
Testbed
Introduction
New user
Existing user
Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact
© Data clustering analysis 2002-2006
Internet

Data Clustering Testbed

Testbed Introduction

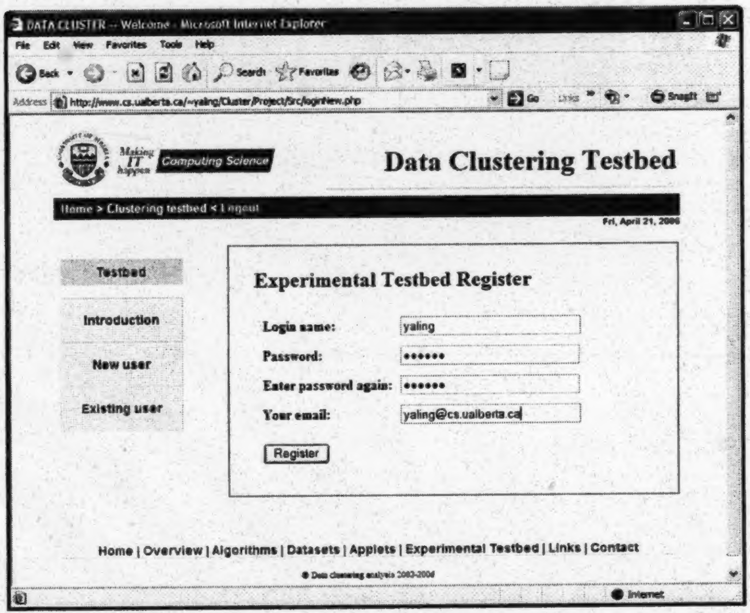
Data Clustering Testbed provides a place where users can experiment different clustering algorithms with selected datasets. So far, eight algorithms have been implemented, some of which are provided by the authors who came up with the algorithms. The implemented algorithms are:

- K-means
- DBSCAN (provided by the authors)
- CURE (algorithm part provided by the authors)
- ROCK (algorithm part provided by the authors)
- CHAMELEON
- CLIQUE
- WaveCluster
- AutoClass

We have some datasets available for you to test with the algorithms. Or you can upload your own datasets to test with different algorithms.

Testbed login

Figure A.5: Clustering testbed introduction page

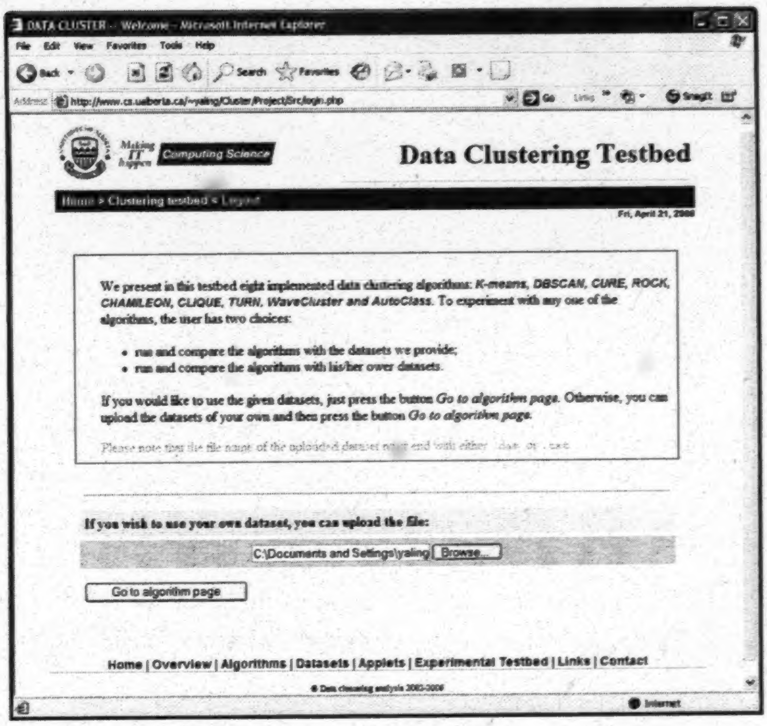

DATA CLUSTER -- Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/loginNew.php
Computing Science
Home > Clustering testbed < Logout
Fri, April 21, 2006
Testbed
Introduction
New user
Existing user
Register
Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact
© Data clustering analysis 2002-2006
Internet

Data Clustering Testbed

Experimental Testbed Register

Login name: yaling
Password: ••••••
Enter password again: ••••••
Your email: yaling@cs.ualberta.ca

Figure A.6: Clustering testbed registration page


DATA CLUSTER -- Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/login.php
Go Links SnagIt
Making IT happen
Computing Science
Data Clustering Testbed
Home > Clustering testbed < Logout
Fri, April 21, 2006
C:\Documents and Settings\yaling
Browse...
Go to algorithm page
Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact
© Data clustering analysis 2002-2006
Internet

We present in this testbed eight implemented data clustering algorithms: ***K-means, DBSCAN, CURE, ROCK, CHAMLEON, CLIQUE, TURN, WaveCluster and AutoClass***. To experiment with any one of the algorithms, the user has two choices:

- run and compare the algorithms with the datasets we provide;
- run and compare the algorithms with his/her ower datasets.

If you would like to use the given datasets, just press the button *Go to algorithm page*. Otherwise, you can upload the datasets of your own and then press the button *Go to algorithm page*.

Please note that the file name of the uploaded dataset must end with either .dat or .txt

If you wish to use your own dataset, you can upload the file:

Figure A.7: Clustering testbed registration validation page

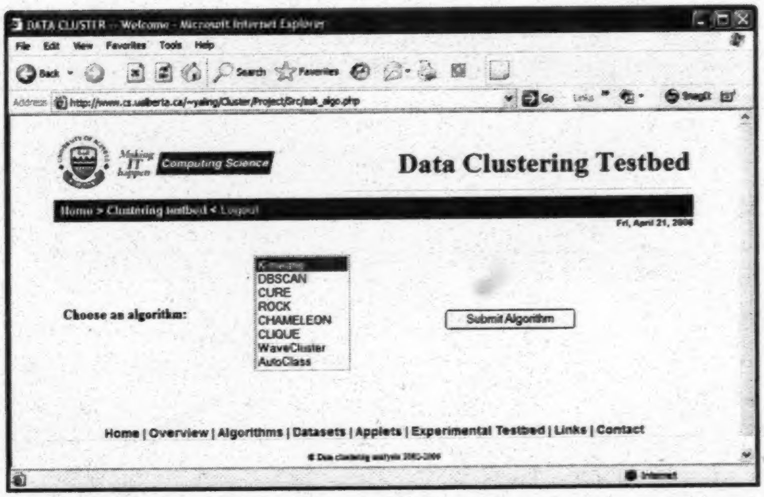
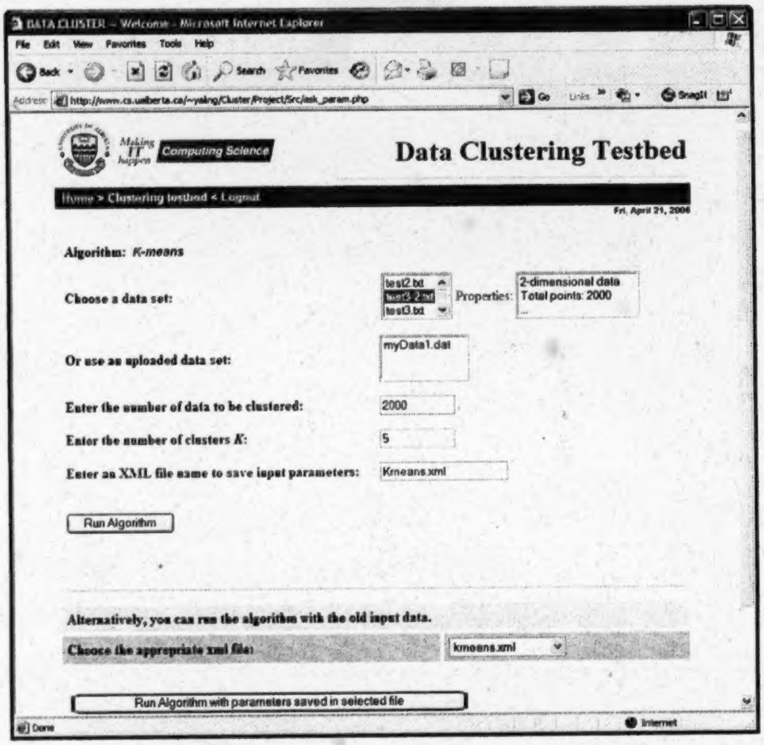

DATA CLUSTER -- Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/ask_algo.php
Go Links SnagIt
Making IT happen
Computing Science
Data Clustering Testbed
Home > Clustering testbed < Logout
Fri, April 21, 2006
Choose an algorithm:
K-means
DBSCAN
CURE
ROCK
CHAMELEON
CLIQUE
WaveCluster
AutoClass
Submit Algorithm
Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact
© Data clustering analysis 2002-2006
Internet

Figure A.8: Clustering testbed algorithm selection page


DATA CLUSTER – Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Go Links SnagIt
Making IT happen
Computing Science
Home > Clustering testbed < Logout
Run Algorithm
Run Algorithm with parameters saved in selected file
Done
Internet

Address: http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/ask_param.php

Data Clustering Testbed

Fri, April 21, 2006

Algorithm: *K-means*

Choose a data set: test2.txt / test3-2.txt / test3.txt — Properties: 2-dimensional data, Total points: 2000 ...

Or use an uploaded data set: myData1.dat

Enter the number of data to be clustered: 2000

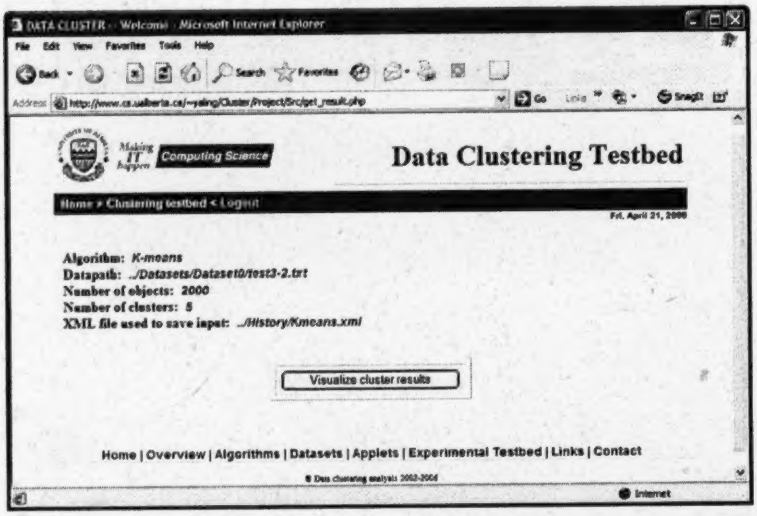
Enter the number of clusters *K*: 5

Enter an XML file name to save input parameters: Kmeans.xml

Alternatively, you can run the algorithm with the old input data.

Choose the appropriate xml file: kmeans.xml

Figure A.9: Clustering testbed algorithm parameter page


DATA CLUSTER – Welcome - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Go Links SnagIt
Making IT happen
Computing Science
Home > Clustering testbed < Logout
Visualize cluster results
Home | Overview | Algorithms | Datasets | Applets | Experimental Testbed | Links | Contact
© Data clustering analysis 2002-2006
Internet

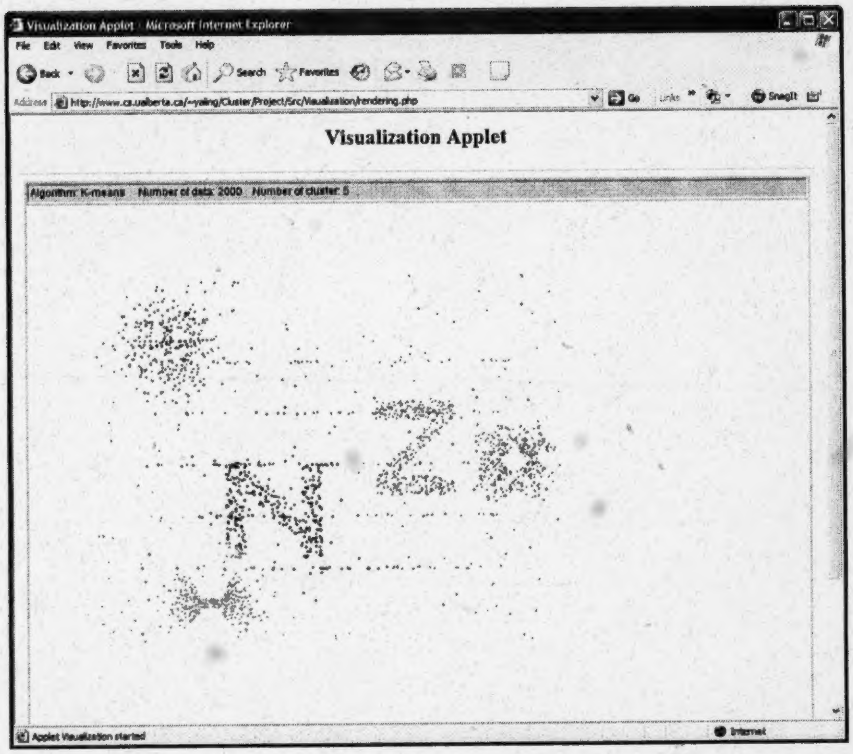
Address: http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/get_result.php

Data Clustering Testbed

Fri, April 21, 2006

Algorithm: *K-means*
Datapath: *../Datasets/Dataset0/test3-2.txt*
Number of objects: *2000*
Number of clusters: *5*
XML file used to save input: *../History/Kmeans.xml*

Figure A.10: Clustering testbed algorithm output page


Visualization Applet - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Address
http://www.cs.ualberta.ca/~yaling/Cluster/Project/Src/Visualization/rendering.php
Go
Links
SnagIt
Internet

Visualization Applet

Algorithm: K-means Number of data: 2000 Number of cluster: 5

Applet Visualization started

Figure A.11: Clustering testbed output presentation page